



Sudam Pullaperuma - 10660248

Tharuka Gunasekara - 10659483

Tanushka Elvitigala - 10663914

Dulaj Walgama - 10659489

## 1.0 EXECUTIVE SUMMARY

The **Aegis Shield Browser Extension** is designed to help users browse the internet safely by identifying harmful websites and protecting personal information. It provides real-time alerts when users visit risky websites and offers tools to block unsafe websites while allowing trusted ones to be accessed without interruptions.

The project was created to address the growing concern of harmful online activities, such as fake websites pretending to be genuine ones, which trick people into sharing personal information like passwords or financial details. To solve this, the extension works in the background to analyze websites and warn users if something seems suspicious or unsafe. Additionally, users can manually add websites they trust or block sites they want to avoid.

To ensure the extension works reliably, the team tested it in multiple stages, improving its performance and fixing any problems along the way. The extension was designed to be simple to use, so anyone, even those without technical knowledge, can benefit from it. It also follows strict privacy rules, ensuring that users' information is not shared or stored unnecessarily.

### Key Achievements:

1. **Real-Time Safety Alerts:** Warns users immediately when visiting risky websites.
2. **Customizable Controls:** Allows users to block unwanted sites and trust specific ones.
3. **Document and File Checking:** Includes a feature for users to upload files and check if they are safe.
4. **Privacy-Focused Design:** Ensures users' data stays private and secure at all times.
5. **Ease of Use:** Designed with a simple interface, making it accessible to everyone.
6. **On-Time Completion:** The project was delivered as planned, meeting all key goals.

The team worked collaboratively, holding regular meetings and sharing progress updates through online tools. They also received guidance from a supervisor to ensure the project stayed on track. Despite some challenges, such as limits on the tools available, the team managed to overcome them without delays.

Looking ahead, there are plans to improve the extension by making it work on other browsers like Firefox and adding smarter tools to make it even more effective. These upgrades will make the extension useful to a wider audience and keep up with changing online risks.

In conclusion, the **Aegis Shield Browser Extension** is a reliable and easy-to-use tool for safe browsing. It not only protects users from harmful websites but also ensures that their information remains private and secure. This report highlights the successful development of the project, showcasing the teamwork and dedication behind it.

## 2.0 TABLE OF CONTENTS

1.0 Executive Summary .....	1
2.0 Table Of Contents .....	2
3.0 Introduction.....	6
4.0 Adherence To Project Proposal .....	7
4.1 Comparison Of Planned Vs. Actual Execution .....	7
4.1.1 Agreed Methods .....	7
4.1.2 Timeline.....	10
4.1.3 Resources.....	14
4.2 Justification Of Variances .....	17
4.3 Supervisor Input And Approval .....	18
4.3.1 Role Of The Supervisor .....	18
4.4 Defined Scope And Implementations .....	20
4.5 Requirement Implementations .....	24
4.5.1 Real-Time Url Monitoring.....	24
4.5.2 Phishing Url Detection .....	25
4.5.3 Basic User Alerts .....	26
4.5.4 Whitelist/Blacklist Management .....	27
4.5.5 Basic Malware Detection.....	29
4.5.6 Email Phishing Detection .....	30
4.5.7 Browser Notifications.....	32
4.5.8 Advanced Machine Learning For Phishing Detection .....	33
4.5.9 Sandbox Integration For File Analysis.....	35
4.5.10 User Friendly Ui, Compliance With Chrome Policies, Secure Data Handling And Lightweight And Efficient Design .....	36
4.5.11 Severity-Based Alerts .....	38
4.5.12 Multi-Browser Compatibility .....	40
4.6 Management Of Requirement Implementations .....	41
5.0 Project Deliverables .....	43
6.0 Used Resources And Components.....	46
6.1 Integration Of Resources And Components.....	47

7.0 Design And Architecture .....	48
7.1 High-Level Workflow And Architecture Diagram .....	48
7.2 Frontend Design .....	48
7.3 Backend Design.....	49
8.0 Compliance With Industry Standards And Best Practices.....	51
8.1 Iso Standards .....	51
8.1.1 Iso/Iec 27001: Information Security Management System .....	52
8.1.2 Iso/Iec 27017: Code Of Practice For Cloud Security .....	52
8.1.3 Iso/Iec 29100: Privacy Framework.....	52
8.2 Gdpr (General Data Protection Regulation).....	53
8.3 Chrome Web Store Developer Program Policies .....	53
8.4 Detailed Explanation Of Alignment.....	54
9.0 Quality Indicators.....	56
9.1 Outputs Of The Product .....	56
9.2 Performance Testing .....	65
9.2.1 Final Version Testing (V4).....	66
9.2.2 Test Cases And Results For Final Version (V4) .....	67
9.2.3 Identified Issues During Final Version Testingversion (V4) .....	80
9.2.4 Future Developments.....	82
10.0 Security And Privacy Management .....	85
11.0 Risk Management .....	88
12.0 Teamwork .....	91
12.1 Excellent Understanding .....	91
12.2 Collaborative Approach To Work.....	92
12.3 Communication Frequency And Clarity .....	95
12.4 Demonstration Of Professionalism .....	98
13.0 Conclusion .....	101
14.0 References .....	102

## TABLE OF FIGURES

Figure 1: Agreed method VS Actual method .....	7
Figure 2: Task Allocation .....	8
Figure 3: Weekly Meetings with Supervisor .....	9
Figure 4: Project Tracking via Shared Folder .....	9
Figure 5: Planned Timeline VS Actual Timeline .....	10
Figure 6: Planned Timeline.....	11
Figure 7: Actual Timeline .....	13
Figure 8:Planned Resources VS Actual Resources .....	14
Figure 9: Design and Architecture .....	50
Figure 10: Evidence 1 .....	56
Figure 11: Evidence 2 .....	56
Figure 12: Evidence 3 .....	57
Figure 13: Evidence 4 .....	57
Figure 14: Evidence 5 .....	58
Figure 15: Evidence 6 .....	58
Figure 16: Evidence 7 .....	59
Figure 17: Evidence 8 .....	59
Figure 18: Evidence 9 .....	60
Figure 19: Evidence 10 .....	60
Figure 20: Evidence 11 .....	61
Figure 21: Evidence 12 .....	61
Figure 22: Evidence 13 .....	62
Figure 23: Evidence 14 .....	62
Figure 24: Evidence 15 .....	63
Figure 25: Evidence 16 .....	63
Figure 26: Evidence 17 .....	64
Figure 27: Evidence 18 .....	64
Figure 28: Evidence 19 .....	65
Figure 29: Evidence 20 .....	65
Figure 30: TEST CASE 1: Real-Time URL Monitoring.....	67
Figure 31: TEST CASE 2: Phishing URL Detection.....	68
Figure 32: TEST CASE 3: Basic User Alerts .....	69
Figure 33: TEST CASE 4: Whitelist/Blacklist Management .....	70
Figure 34: TEST CASE 5: Basic Malware Detection .....	71
Figure 35: TEST CASE 6: Email Phishing Detection .....	72
Figure 36: TEST CASE 7: Browser Notifications.....	73
Figure 37: TEST CASE 8: User-Friendly Interface .....	74
Figure 38: TEST CASE 9: Severity-Based Alerts .....	75

Figure 39: TEST CASE 10: Advanced Machine Learning for Phishing Detection .....	76
Figure 40: TEST CASE 11: Multi-Browser Compatibility .....	77
Figure 41: TEST CASE 12: Sandbox Integration for File Analysis.....	78
Figure 42: Evaluation of Versions .....	80
Figure 43: Security and Privacy Management.....	85
Figure 44: Task Allocation Among Team Members .....	92
Figure 45: Shared folder environment .....	93
Figure 46: MS Teams used to share updates .....	93
Figure 47: Practicing the presentation using Zoom meetings.....	94
Figure 48: Communication VIA MS Teams.....	95
Figure 49: Reviewing one of the documents .....	96
Figure 50: Giving editing access to the team on one of the deliverables through Personal mail .	97
Figure 51: Inviting team members to a GitHub repository .....	97
Figure 52: Sharing Panopto recorded demo video to review.....	97
Figure 53: Supervisor Meeting Recordings .....	98

## LIST OF TABLES

Table 1: Planned Resources .....	16
Table 2: Proposed Scope.....	21
Table 3: Summary of Implementation .....	23
Table 4: Agreed deliverables .....	43
Table 5: Delivered Deliverables .....	45
Table 6: Used Resources and Components.....	47
Table 7: Overview of Final Version Testing .....	66
Table 8: Evaluation of Versions .....	79
Table 9: Expected Risks.....	89

### 3.0 INTRODUCTION

The prevalence of cyber threats, particularly phishing attacks, has made online security a critical concern in today's digital landscape. Phishing, a deceptive practice that tricks users into revealing sensitive information by impersonating legitimate entities, poses significant risks to individuals and organizations. With the increasing sophistication of such attacks, the need for accessible and reliable security tools has become more pressing. Recognizing this challenge, the **Aegis Shield Phishing Detection Extension** was developed to enhance browser security by providing real-time threat detection and user-friendly features.

The internet's integral role in our daily lives has amplified the importance of safeguarding online activities. Existing security solutions, while effective, often lack ease of use or fail to cater to non-technical users. Our project bridges this gap by offering a Chrome browser extension that combines real-time phishing detection, machine learning capabilities, and a simple, intuitive interface. By leveraging advanced tools such as VirusTotal API and machine learning models, the extension ensures robust protection against phishing and malware threats.

This report documents the comprehensive execution of the project, covering all phases from planning and development to testing and delivery. It provides a detailed analysis of the implemented features, including Real-Time URL Monitoring, Phishing URL Detection, and Whitelist/Blacklist Management. Additionally, the report highlights the team's commitment to adhering to international standards such as **ISO/IEC 27001**, **GDPR**, and **Chrome Web Store Developer Policies**, ensuring security, privacy, and compliance throughout the project.

A proactive approach to **risk management** ensured that anticipated challenges, such as API limitations and timeline adjustments, were effectively addressed. Furthermore, rigorous **testing and evaluation** validated the extension's functionality and reliability, while collaborative efforts among team members ensured efficient execution of tasks. The report also outlines future developments, such as expanding multi-browser compatibility and integrating dynamic machine learning models, to broaden the extension's scope and impact.

In summary, the **Aegis Shield Phishing Detection Extension** not only meets its objectives but also demonstrates innovation, collaboration, and adherence to best practices. This report serves as a comprehensive account of the project's journey, showcasing the technical and managerial expertise that contributed to its success while emphasizing its value as a reliable tool for enhancing online security.

## 4.0 ADHERENCE TO PROJECT PROPOSAL

This section provides an in-depth evaluation of how closely the project execution aligned with the objectives, methods, and resources outlined in the original project proposal. It includes a comparison of planned versus actual execution, discussing adjustments made during the project and their impact on deliverables. Additionally, it evaluates the management of requirement implementations, detailing the prioritization of tasks, integration of supervisor feedback, and how risks were mitigated. By examining these aspects, this section demonstrates the team's commitment to delivering a product that adheres to the agreed scope and quality standards.

### 4.1 COMPARISON OF PLANNED VS. ACTUAL EXECUTION

This section evaluates how the project's actual execution compared to the planned timeline and methodology. It examines the alignment between proposed goals and deliverables versus the actual outcomes, highlighting any deviations and their resolutions. The discussion includes how tasks were prioritized and adjusted, the impact of changes to the original plan, and how these adjustments ensured the successful completion of the project while adhering to its scope and objectives. Additionally, it covers the use of tools like the Project Tracker and supervisor feedback to manage tasks and timelines effectively.

#### 4.1.1 AGREED METHODS

During the initial stages of the project, our team faced the critical task of selecting the most suitable development methodology to guide our work. Recognizing that the choice of methodology would significantly influence the project's success, we conducted a brief research study on various approaches, including Waterfall, Agile, and Hybrid models.

After evaluating these methods based on factors such as flexibility, iterative development, and stakeholder involvement, we identified Agile as the optimal choice. Agile's emphasis on incremental progress, continuous feedback, and adaptability aligned perfectly with our project goals, enabling us to respond effectively to evolving requirements and unforeseen challenges.


Planned		Actual
Use sprint-based Agile		Divided tasks by expertise, not sprints.
Weekly feedback meetings to refine the project.		Held 8 of 9 planned weekly meetings.
Formal project management tool for project tracking		Used a shared folder for real-time updates.
Implement feedback in subsequent sprints		Addressed feedback immediately without sprints.

Figure 1: Agreed method VS Actual method



## Planned Approach

As outlined in the original project proposal, the development methodology chosen for this project was the **Agile Approach**. This methodology was selected for its ability to provide flexibility, iterative progress, and continuous feedback, enabling the team to adapt to evolving requirements and deliver a high-quality final product. The key principles highlighted in the proposal were:

- **Flexibility:** The ability to adjust to changes in requirements or scope during the development process.
- **Iterative Progress:** Delivering the project in smaller, manageable increments to ensure steady progress.
- **Feedback from Supervisor:** Actively seeking input from the project supervisor at every stage to refine features and address any issues proactively.

The Agile approach emphasized collaboration, adaptability, and a focus on delivering functional components throughout the project lifecycle.

## Actual Execution

During the project, the team adhered to the Agile methodology as planned, fully implementing its principles with minor modifications to suit the project needs. The following practices were adopted to align with the methodology:

### 1. Task Assignment and Individual Contributions

- Instead of dividing the project into traditional tasks were distributed among team members based on individual expertise and availability. Each member was responsible for completing their assigned portion of the project, ensuring steady progress.

Task No	Task	Required Resources	Estimated Dates	Estimated Deadline	Actual Dates	Actual Deadline	Responsible Person	Notes	Status
1	Define Project Scope	Project management software (MS Excel)	3	25/11/2024	2	25/11/2024	Tanushka	Schedule a meeting with the supervisor for feedback on the project scope.	Completed
2	Set Up Development Environment	VS Code, Chrome Developer Tools, Node.js, APIs	2	30/11/2024	2	30/11/2024	Dulaj	Confirm installation and setup with the supervisor.	Completed
3	Familiarize with Chrome Extension Guidelines	Chrome Developer Guide, API documentation	2	30/11/2024	1	30/11/2024	Tharuka	Supervisor to verify understanding of guidelines.	Completed
4	Schedule the first meeting with supervisor	Ms Teams, Outlook	1	31/11/2024	1	31/11/2024	Sudam	Confirm the supervisor's availability to have a meeting	Completed
5	1st supervisor meeting	Ms Teams	1	2/12/2024	1	2/12/2024	Team	Discuss the project idea with the supervisor.	Completed
6	Create a project Proposal	MS Word	5	8/12/2024	4	8/12/2024	Team	Finalize the scope and complete the project proposal	Completed
7	Schedule the Second meeting with supervisor	Ms Teams, Outlook	1	9/12/2024	1	9/12/2024	Sudam	Confirm the supervisor's availability to have a meeting	Completed
8	2nd supervisor meeting to get the approval for the proposal	Ms Teams, MS Word	1	10/11/2024	1	10/11/2024	Team	Got the supervisor approval	Completed
9	Get the feedback from Miss Ann for the proposal	MS word	1	13/12/2024	1	13/12/2024	Team	Got the feedback	Completed
10	Finalize the proposal based on the feedback	MS Word	2	13/12/2024	1	13/12/2024	Team	Finalized the Proposal	Completed
11	Create and sign the Group Contract	MS Word	2	14/12/2024	1	14/12/2024	Team	Created the team agreement	Completed
12	Project Proposal and Group Contract Submission	Canvas	1	14/12/2024	1	14/12/2024	Sudam	Submitted both Proposal and group contract	Completed
13	Setting Up a Repository on GitHub to share the Codes	GitHub	1	14/12/2024	1	14/12/2024	Tanushka	All members are given access to the Repo	Completed
14	Design the Basic UI, Prototype	Adobe Illustrator, HTML, CSS	3	14/12/2024	3	14/12/2024	Tharuka	Discuss the overall structure and get the ideas	Completed
15	Conduct a research UI design best practices	Figma, Illustrator, Canva	2	16/12/2024	2	16/12/2024	Dulaj	Schedule a discussion on findings	Completed
16	3rd Supervisor Meeting	MS Teams	1	16/12/2024	1	16/12/2024	Team	Discuss the progress	Completed
17	Design wireframes and prototypes	Figma, Illustrator, Canva	2	18/12/2024	2	18/12/2024	Sudam	Share designs with the supervisor for feedback	Completed
18	Develop core UI elements	VS code and Anaconda	1	20/12/2024	1	20/12/2024	Tharuka	Schedule a discussion on backend integration challenges	Completed
19	Conduct a research on API	Vinustotal API	1	21/12/2024	1	21/12/2024	Tanushka	Share API research outcomes	Completed
20	4th Supervisor Meeting	Ms Teams	1	23/12/2024	1	23/12/2024	Team	Review progress	Completed
21	Dataset preprocessing and cleaning	Jupyter, VS code	3	27/12/2024	3	27/12/2024	Tanushka	Share preprocessing steps	Completed
22	Split dataset into training and testing sets	Jupyter, VS code	2	29/12/2024	2	29/12/2024	Sudam	Schedule a session to discuss data splits and any concerns	Completed

Figure 2: Task Allocation

## 2. Weekly Meetings with Supervisor

- Weekly meetings were conducted with the project supervisor to discuss progress, challenges, and next steps. A total of 9 meetings were planned, with 8 meetings completed as of January 22, 2025.
- These meetings provided an opportunity to receive feedback, resolve issues, and ensure alignment with the project objectives.

Meeting Number	Date	Link for the recording
1	Monday, December 2, 2024	<a href="https://edithcowanuni-my.sharepoint.com/:v:/g/">https://edithcowanuni-my.sharepoint.com/:v:/g/</a>
2	Tuesday, December 10, 2024	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
3	Monday, December 16, 2024	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
4	Monday, December 30, 2024	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
5	Sunday, January 5, 2025	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
6	Monday, January 6, 2025	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
7	Tuesday, January 14, 2025	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>
8	Wednesday, January 22	<a href="https://edithcowanuni-my.sharepoint.com/perso">https://edithcowanuni-my.sharepoint.com/perso</a>

Figure 3: Weekly Meetings with Supervisor

### 3. Progress Tracking via Shared Folder

- To maintain transparency and streamline collaboration, a shared folder was created.
- Completed tasks and deliverables were uploaded to this folder, ensuring that all team members, the supervisor, and the lecturer could access and review the progress in real time. This process reduced the need for frequent internal meetings while maintaining visibility and accountability.

Tharuka GUNASEKARA > SRI-CSG3101.2 - phishing detection extension

















	Name	Modified	Modified By	File size	Sharing
	Approved project proposal	December 11, 2...	Sudam PULLAPERU	1 items	 Shared
	Group Contract	December 14, 2...	Sudam PULLAPERU	1 items	 Shared
	Meeting Details	January 6	Sudam PULLAPERU	1 items	 Shared
	Meeting Recordings	January 6	Sudam PULLAPERU	1 items	 Shared
	Project Tracker	December 9, 20...	Sudam PULLAPERU	1 items	 Shared
	Timeline	December 9, 20...	Sudam PULLAPERU	1 items	 Shared
	applied-01.jpg	December 9, 20...	Tharuka GUNASEK	1.05 MB	 Shared
	Project draft.docx	December 9, 20...	Sudam PULLAPERU	17.9 KB	 Shared

Figure 4: Project Tracking via Shared Folder

#### 4. Incorporating Supervisor Feedback

- Feedback from the supervisor during weekly meetings was consistently integrated into the project. For instance, suggestions on improving the user interface and refining URL detection methods were implemented promptly.

The adherence to the Agile methodology, as planned in the proposal, was critical to the project's success. Although the traditional sprint-based approach was adjusted to individual task assignments and weekly meetings, the core principles of collaboration, feedback, and iteration were consistently applied. These practices enabled the team to complete the project effectively while maintaining transparency and alignment with the supervisor's expectations.

##### 4.1.2 TIMELINE

The team encountered a situation early in the project where a clear plan was essential to ensure smooth progress. As an action, we developed a detailed timeline by breaking down the work into smaller, manageable tasks and assigning them to respective phases. This structured approach not only provided clarity on responsibilities but also allowed us to track progress effectively. As a result, the team was able to adhere to the timeline and successfully deliver the project on the planned date, January 25, 2025.

While the team followed the planned timeline closely, minor adjustments were made to accommodate task dependencies and ensure quality deliverables. These changes, such as extending testing phases or refining documentation tasks, did not impact the overall project schedule, and the final delivery date of January 25, 2025, was successfully met.

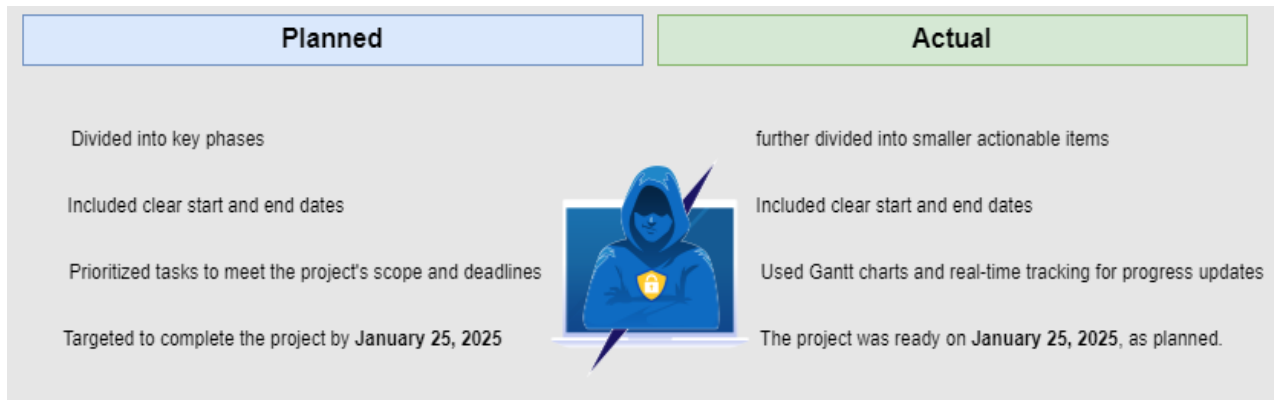


Figure 5: Planned Timeline VS Actual Timeline

## Planned Timeline

The project's timeline was carefully structured and represented using the Gantt chart, ensuring clear planning and scheduling of all tasks. The timeline divided the project into multiple phases, with specific start and end dates for each activity.

Key phases outlined in the Gantt chart included:

1. **Project Initialization (Nov 23 - Nov 26, 2024):**
  - Activities such as requirement analysis and planning were completed to lay the groundwork for the project.
2. **Learning Tools and APIs (Nov 27 - Dec 3, 2024):**
  - The team focused on understanding the VirusTotal API and Chrome Extensions, essential for implementing core features.
3. **Core Feature Development (Dec 4 - Dec 21, 2024):**
  - Development of critical components such as Real-Time URL Monitoring, Phishing URL Detection, and Basic Malware Detection occurred during this phase.
4. **Testing and Debugging (Jan 4 - Jan 7, 2025):**
  - Internal testing, debugging, and bug-fixing activities were conducted to ensure functionality and performance.
5. **Documentation (Jan 8 - Jan 14, 2025):**
  - User guides, developer manuals, and requirement reports were prepared.
6. **Final Presentation (Planned: Jan 22 - Jan 25, 2025):**
  - Preparation and delivery of the final presentation to conclude the project.

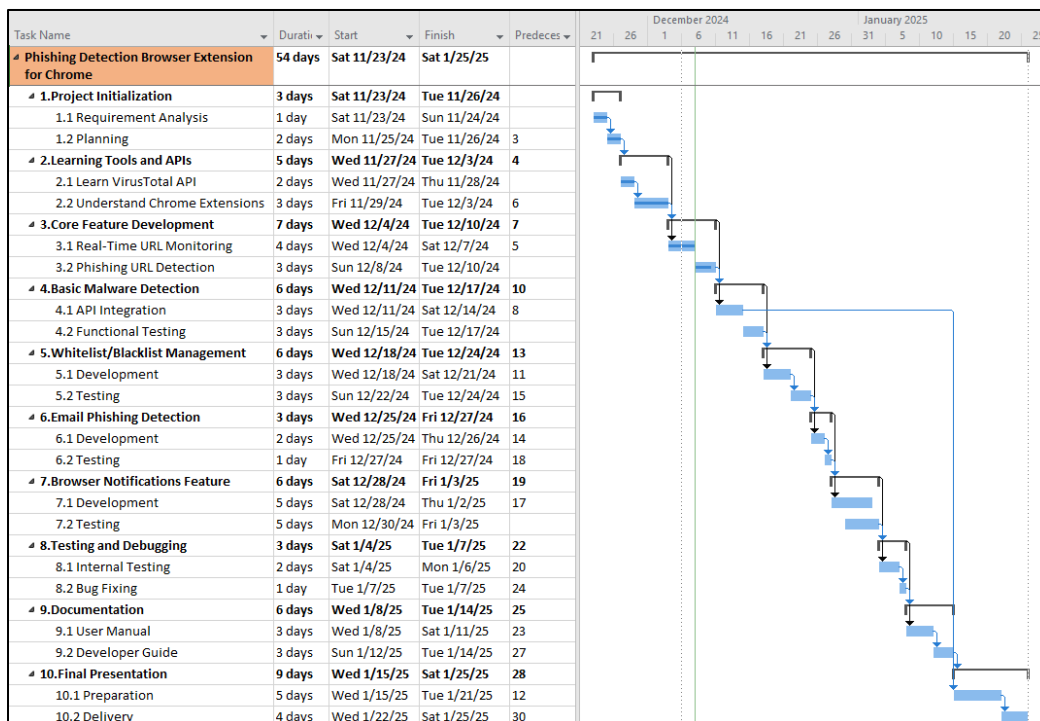


Figure 6: Planned Timeline

## Actual Execution

During the project's execution, the team realized that further breaking down the planned timeline into more granular tasks would enhance clarity and manageability. To address this, we developed an **updated timeline**, which served as a detailed work breakdown structure. The project spanned **54 days** from **November 23, 2024, to January 25, 2025**, and successfully met the planned delivery deadline.

This decision provided several benefits:

1. **Improved Clarity:** Team members clearly understood their specific responsibilities and deliverables.
2. **Enhanced Collaboration:** The detailed breakdown allowed everyone to track progress in real-time, fostering a sense of accountability.
3. **Flexibility:** Breaking down tasks made it easier to adjust timelines when challenges arose, ensuring steady progress toward the final goal.

The timeline was divided into ten major phases, further broken down into sub-tasks for streamlined execution:

1. **Project Initialization:**
  - This phase included defining the project scope, creating the proposal, scheduling meetings with the supervisor, and ensuring feedback loops.
  - Additional sub-tasks like creating a group contract and setting up a GitHub repository for collaboration ensured project readiness.
2. **Learning Tools and APIs:**
  - The team prioritized understanding VirusTotal API, Chrome Extension guidelines, and development environment setup.
  - Conducted research on UI/UX best practices and finalized wireframes and prototypes.
3. **Core Feature Development:**
  - Core features, such as **Real-Time URL Monitoring, Phishing URL Detection, and Basic Malware Detection**, were implemented and tested iteratively.
  - UI prototypes and notifications for flagged URLs were seamlessly integrated during this stage.
4. **Whitelist/Blacklist Management and Email Phishing Detection:**
  - This stage included the implementation of management features and the integration of machine learning models for phishing detection.
  - Dataset preprocessing, cleaning, and testing ensured the models met quality standards.
5. **Browser Notifications and Testing:**
  - Notifications for severity-based alerts were developed alongside functional testing of browser notifications.
  - Internal testing phases uncovered key bugs, which were resolved in the bug-fixing stage.

## 6. Documentation:

- User manuals, developer guides, and reports (e.g., testing, evaluation, and requirements reports) were prepared and finalized to accompany the deliverables.

## 7. Final Presentation:

- Preparation and delivery tasks included PPT creation, combining code files, and rehearsals to ensure polished delivery.

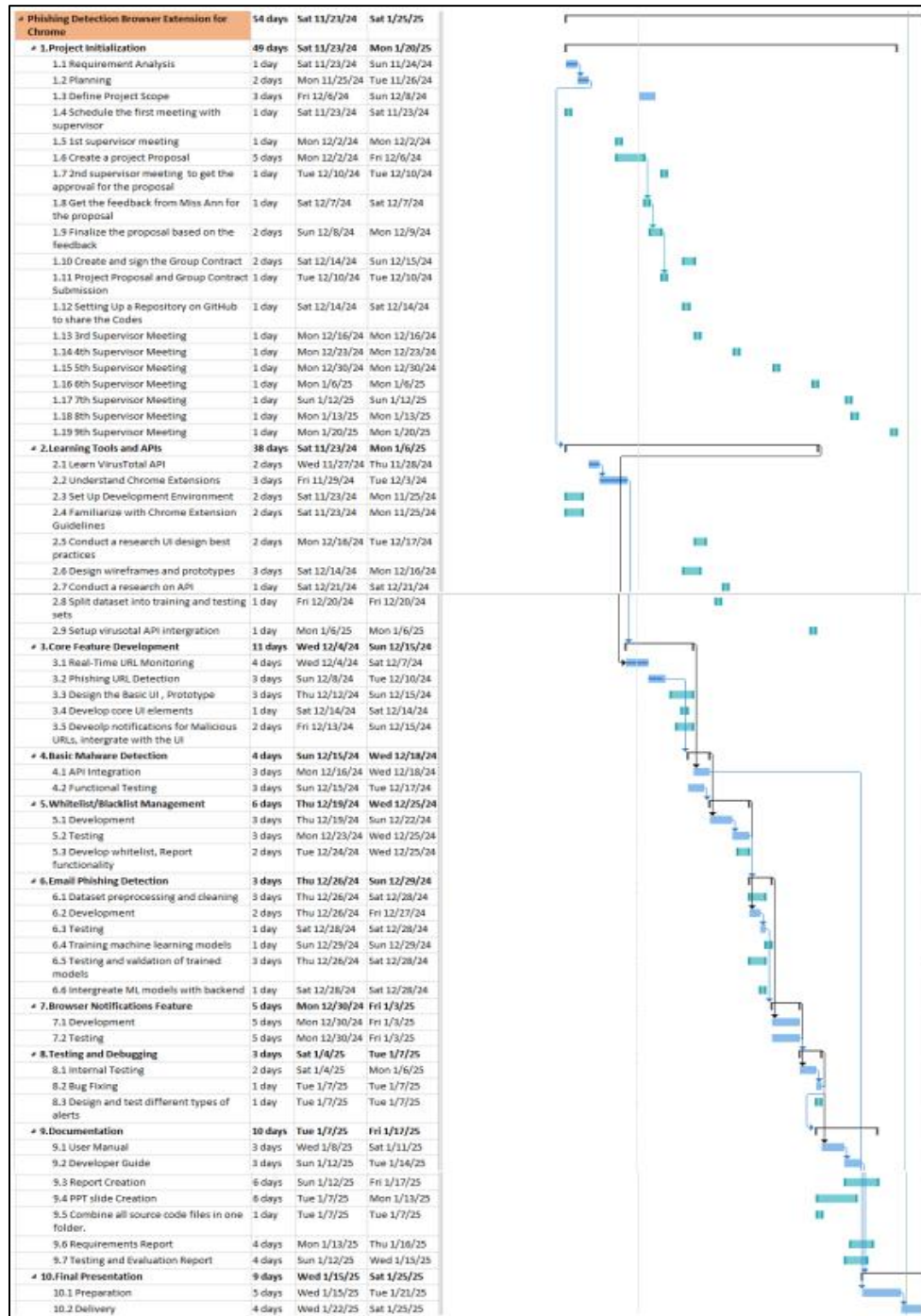


Figure 7: Actual Timeline

As a result of condensing the actual timeline into one frame, its quality may appear reduced; for a clearer view, please refer to the submitted zip folder where the full actual timeline is included or follow this link to download it [Actual Timeline](#).

### **Justification for Actual Timeline Approach**

The use of a detailed actual timeline approach was crucial for ensuring the project stayed on track while adapting to real-time challenges and feedback. By breaking down the tasks into granular activities, the team was able to monitor progress more effectively and address dependencies or delays immediately. The integration of a Gantt chart with detailed sub-tasks provided a clear visualization of project milestones, ensuring alignment with the planned timeline. Additionally, the incorporation of supervisor feedback and the agile methodology allowed the team to prioritize critical features while accommodating flexibility for adjustments. This approach not only facilitated better resource allocation but also ensured that every team member was aware of their responsibilities, contributing to the timely and successful delivery of the project on **January 25, 2025**.

### **4.1.3 RESOURCES**

We faced a situation where it was essential to finalize the tools and resources required to complete the project effectively. As an action, we conducted research on existing browser extensions and their development processes. Based on the insights gained, we carefully selected the tools, APIs, and frameworks that best aligned with our project needs, ensuring a robust and efficient development process.

The resources outlined in the project proposal were strictly followed, with no deviations or additional tools introduced during the project. This consistency ensured that the project remained aligned with the original plan. The effective utilization of these resources contributed significantly to the successful development, testing, and delivery of the **Phishing Detection Browser Extension for Chrome**, showcasing the team's efficiency and adherence to the proposed resource plan.

Planned	Actual
Use two Dataset from Kaggle	Used Kaggle datasets as planned
Integrate VirusTotal API	Fully integrated VirusTotal API
Visual Studio Code, Chrome Developer Tools, and Anaconda for development	Used Visual Studio Code, Chrome Developer Tools, and Anaconda as planned
HTML, CSS, JavaScript for front-end	Used as planned
Python and Flask for back-end	Used as planned
Use GitHub for version control	Managed code with GitHub as planned.

*Figure 8:Planned Resources VS Actual Resources*



## Planned Resources

As outlined in the project proposal, a detailed list of tools, datasets, programming frameworks, and hardware requirements was prepared to support the successful execution of the project. These resources were selected to ensure the development, testing, and delivery phases were conducted efficiently. The categories included datasets, APIs, development tools, programming frameworks, collaboration tools, documentation tools, and hardware.

Category	Item	Purpose	Source	Access Method
Datasets	Phishing Site URLs	Training and validating phishing detection logic	Kaggle	Download using university-provided corporate email.
	Malicious URLs Dataset	Enhancing detection of malicious patterns	Kaggle	Download using university-provided corporate email.
APIs	VirusTotal API	Real-time phishing and malware URL detection	VirusTotal API	Register with university-provided corporate email and obtain API key.
Development Tools	Visual Studio Code	Code editor for extension development	Visual Studio Code	Free download from official website.
	Chrome Developer Tools	Testing and debugging the extension	Built into Google Chrome	Access through the Chrome browser.
	Anaconda	Environment for Python-based development and dependency management	Anaconda.com	Download and install from official website using personal or institutional resources.
Programming Languages and Frameworks	HTML, CSS, JavaScript	Building the Chrome extension's front-end	Open Source	Access and write code directly in Visual Studio Code.
	JSON	Data handling and integration	Open Source	Incorporated as part of JavaScript and API responses.
	Python and Flask	Back-end development and API integration	Open Source	Install Python and Flask via Python.org and pip.
Collaboration Tools	GitHub	Version control and team collaboration	GitHub	Use free account with personal emails.
Documentation Tools	Microsoft Word and Google Docs	Preparing user manuals and developer guides	Microsoft Office or Google Docs	Use free or institutional licenses.
	Canva or PowerPoint	Creating presentation materials	Canva or Microsoft Office	Access free Canva plans or use institutional



				Microsoft PowerPoint.
	Adobe Illustrator	Designing visual assets and diagrams for documentation and presentations	Adobe.com	Institutional or personal license required for access.
Hardware	Laptops/Desktops	Development and testing	Personal	Provided by team members or through university resources.
	Internet Connection	Access APIs, download datasets, and collaborate	Personal or institutional	Access via personal or university networks.

Table 1: Planned Resources

### **Resources Used**

The team adhered to the planned resources, utilizing every item listed in the project proposal. Below is an overview of how these resources were implemented during the project:

**Datasets:** Two datasets were utilized during the project:

- **Phishing Site URLs** from Kaggle were employed to train and validate the phishing detection logic. These datasets were accessed using university-provided corporate email accounts.
- Similarly, the **Malicious URLs Dataset** was used to enhance the detection of malicious patterns. Both datasets were integral to ensuring accurate detection capabilities.

**APIs:** The **VirusTotal API** played a critical role in real-time phishing and malware URL detection. The API key was obtained through a corporate university email, ensuring secure and effective integration. The API's functionality was essential for analyzing URLs and detecting potential threats.

**Development Tools:** Several tools were utilized for the project's development phase.

- **Visual Studio Code** was used as the primary code editor for building the Chrome extension. It was downloaded free from the official website.
- **Chrome Developer Tools**, built into the Google Chrome browser, were used extensively for testing and debugging.
- **Anaconda** provided an environment for Python-based development and dependency management. It was installed via the official website, using institutional or personal resources.

**Programming Languages and Frameworks:** The project leveraged a variety of programming languages and frameworks:

- **HTML, CSS, and JavaScript** were used for the front-end of the extension, while **JSON** was incorporated for data handling and integration.
- For back-end development, **Python and Flask** were employed, enabling effective API integration. These tools were downloaded and installed via their respective official sources.

**Collaboration Tools:** **GitHub** was used extensively. The team leveraged free GitHub accounts to maintain code repositories and track updates efficiently.

**Documentation Tools:** The preparation of user manuals, developer guides, and presentation materials involved multiple tools:

- **Microsoft Word and Google Docs** were used to draft and finalize documentation. These tools were accessed through free or institutional licenses.
- Presentation materials were created using **Canva** and **Microsoft PowerPoint**, ensuring high-quality visuals and professional designs.
- **Adobe Illustrator** was utilized to design diagrams and visual assets for documentation, leveraging institutional licenses.

## 4.2 JUSTIFICATION OF VARIANCES

The project was executed strictly within the boundaries of the originally defined scope, with no changes made to the planned objectives or deliverables. From the outset, the project scope emphasized the development of a robust Chrome browser extension capable of detecting phishing and malicious URLs, providing real-time alerts, and enabling whitelist/blacklist management. These core features, along with supplementary functionalities like Advanced Machine Learning for Phishing Detection and Sandbox Integration for File Analysis, were all implemented exactly as outlined in the project proposal.

The team ensured that every aspect of the defined scope was thoroughly addressed, from technical features to user interface design. Despite minor adjustments to task timelines during execution, the scope itself remained unchanged. All planned components, including real-time URL monitoring, integration with the VirusTotal API, and functionality for detecting and managing phishing threats, were delivered as initially envisioned.

This strict adherence to the original scope was made possible through detailed planning, regular team updates, and consistent feedback from the project supervisor. By following the proposed timeline and resource plan, the team successfully avoided any scope creep or deviation, ensuring that the final product fully aligned with the objectives set out at the beginning of the project. The deliverables, as defined in the proposal, were delivered as planned, meeting both functional and non-functional requirements without compromise.

## 4.3 SUPERVISOR INPUT AND APPROVAL

The supervisor's proactive involvement and commitment to the project were key factors in its success. His input not only helped the team address technical and functional requirements but also instilled a sense of discipline and professionalism.

### 4.3.1 ROLE OF THE SUPERVISOR

The supervisor played a pivotal role in the success of this project, providing valuable guidance and continuous support throughout the development process. From the project's inception to its completion, the supervisor's input was instrumental in shaping the direction and ensuring the achievement of the project's objectives. His feedback and advice were not only constructive but also essential in overcoming challenges and refining the deliverables to meet the highest standards.

The team conducted **eight meetings with the supervisor** as of **January 22, 2025**, as part of the project plan. Each meeting served as a platform for idea sharing, progress reviews, and constructive discussions. The supervisor's ability to provide clear insights and actionable recommendations ensured that the project stayed aligned with its scope and timeline.

Key aspects of his involvement include:

- **Guidance and Expertise:** The supervisor's expertise in the field was evident during every interaction. He guided the team on technical challenges, such as optimizing the phishing detection algorithms, ensuring compliance with Chrome Web Store policies, and improving the user interface for better usability. His advice on leveraging tools like the VirusTotal API and best practices for testing was invaluable in delivering a robust final product.
- **Feedback on Deliverables:** Regular feedback during meetings allowed the team to identify gaps and improve the quality of deliverables. For example, early feedback on the user interface led to enhancements in navigation and accessibility, making the extension more intuitive for end users. His comments on documentation ensured clarity and professionalism in the user manual and developer guide.
- **Idea Sharing and Collaboration:** The meetings fostered a collaborative environment where ideas were exchanged freely. The supervisor encouraged critical thinking and helped refine the project's features to align with real-world requirements. His ability to ask thought-provoking questions and suggest alternative approaches enriched the overall development process.
- **Project Alignment and Motivation:** The supervisor consistently ensured that the project remained aligned with its objectives and timeline. His encouragement and motivational support were crucial during challenging phases, such as testing and debugging.

### 4.3.2 DIRECTED CHANGES AND THEIR IMPACT

During the course of the project, the supervisor provided key advice that significantly influenced the final deliverables. One of the most impactful suggestions was to revisit the features listed in the "Won't Have" section of the proposed scope: Multi-Browser Compatibility and Sandbox Integration for File Analysis. This recommendation came after the team successfully completed the core features ahead of schedule, creating an opportunity to explore additional functionalities to enhance the extension's value.

#### Directed Changes by the Supervisor

- **Multi-Browser Compatibility:** The supervisor suggested that expanding the extension's functionality to other browsers, such as Firefox and Edge, would increase its usability and reach. Acting on this advice, the team allocated resources to implement this feature, focusing on ensuring the extension could function in multiple browser environments. Although the development process faced some technical challenges, the team successfully completed **70% of the Multi-Browser Compatibility implementation**. This partial implementation provides a strong foundation for future enhancements, making the extension adaptable for other browsers.
- **Sandbox Integration for File Analysis:** Another suggestion was to implement a sandbox environment for file analysis, enabling users to scan and evaluate files for potential threats. With this feature, users can upload files to be analyzed in a secure, isolated environment, enhancing the extension's malware detection capabilities. The team embraced this recommendation and successfully implemented the feature **100%**, delivering a fully functional sandbox integration. This addition added a significant layer of security and extended the extension's utility beyond phishing detection to malware protection.

#### Impact of the Directed Changes

The supervisor's advice to implement these features resulted in a considerable enhancement to the extension's overall value and functionality.

- **Expanded Usability:** The partial implementation of Multi-Browser Compatibility ensures that the extension has the potential to serve users across multiple browsers, making it more accessible and appealing to a wider audience.
- **Enhanced Security:** The Sandbox Integration for File Analysis improved the extension's ability to safeguard users from malicious files, further solidifying its position as a comprehensive security tool.
- **Added Value:** Incorporating these advanced features demonstrated the team's ability to go beyond the proposed scope and utilize available time efficiently. This not only added technical depth to the project but also showcased the team's commitment to delivering a high-value product.

## 4.4 DEFINED SCOPE AND IMPLEMENTATIONS

The project's scope was clearly defined at the outset using the **MoSCoW prioritization framework**, categorizing features into **Must-Have**, **Should-Have**, **Could-Have**, and **Won't-Have** requirements. The **Must-Have** requirements were identified as core deliverables critical to the functionality and success of the phishing detection browser extension. The project aimed to achieve all Must-Have and non-functional requirements to deliver a fully operational and secure extension, while the optional features from other categories were deprioritized due to time and resource constraints.

The key objectives included implementing features such as **Real-Time URL Monitoring**, **Phishing URL Detection**, **Basic User Alerts**, **Whitelist/Blacklist Management**, **Basic Malware Detection**, **Email Phishing Detection**, and **Browser Notifications**, alongside ensuring the extension's lightweight design, user-friendliness, compliance with Chrome policies, and secure data handling.

Priority	Requirement Type	Requirement	Description
Must-Have	Functional	Real-Time URL Monitoring	Monitor all visited URLs in the browser and analyze them for phishing or malware indicators using VirusTotal API.
		Phishing URL Detection	Detects and flag suspicious or malicious URLs based on API analysis results.
		Basic User Alerts	Provide real-time notifications or pop-ups when malicious URLs are detected, with actionable options like "Block" or "Proceed with caution."
		Whitelist/Blacklist Management	Allow users to add trusted websites (whitelist) and blocked websites (blacklist).
		Basic Malware Detection	Use VirusTotal API to analyze URLs for potential malware threats.
		Email Phishing Detection	Allow users to paste email content or headers for analysis and detect phishing indicators in embedded links or sender details
		Browser Notifications	Send categorized browser notifications with severity levels, such as safe, suspicious or malicious.
	Non-Functional	Lightweight and Efficient Design	The extension must have a lightweight design to ensure minimal impact on browser performance
		User-Friendly Interface	Provide an intuitive interface that is accessible to non-technical users.

		Compliance with Chrome Policies	Adhere to Chrome Web Store policies for extension development and security.
		Secure Data Handling	Ensure user data is handled securely, complying with data protection standards.
Should-Have	Functional	Severity-Based Alerts	Provide additional context in alerts, such as the threat level and recommended actions for users.
		Email Content Parsing	Automatically parse and analyze structured email content for embedded phishing links.
Could-Have	Functional	Advanced Machine Learning for Phishing Detection	Integrate pre-trained models to identify phishing patterns and implement basic machine learning models for improved detection accuracy.
		Heuristic URL Analysis	Add support for detecting suspicious patterns in URLs, such as typosquatting, IP-based URLs, and uncommon TLDs.
		Customizable User Settings	Enable users to adjust detection sensitivity (e.g., strict vs. moderate modes) and toggle features like malware scanning or heuristic analysis.
Won't-Have	Functional	Multi-Browser Compatibility	Extend the extension to support browsers like Firefox, Edge, and Brave.
		Sandbox Integration for File Analysis	Add a feature to upload and scan files for malware using APIs like VirusTotal.

Table 2: Proposed Scope

## Implemented Features

The project achieved significant success in implementing all planned deliverables within the Must-Have category, while also making progress on some Should-Have and Could-Have features. Below is a detailed breakdown of the implementation:

### 1. **Must-Have Requirements (100% Implemented)**

- **Real-Time URL Monitoring:** Successfully implemented to monitor all visited URLs in the browser, leveraging the VirusTotal API to detect phishing and malware indicators in real-time.
- **Phishing URL Detection:** Fully functional feature to detect and flag suspicious or malicious URLs using API analysis results.
- **Basic User Alerts:** Real-time notifications and pop-ups were implemented to warn users of malicious URLs, with actionable options like "Block" or "Proceed with Caution."
- **Whitelist/Blacklist Management:** Allowed users to add trusted or blocked websites, enabling customization of URL handling.

- **Basic Malware Detection:** Integrated with VirusTotal to scan URLs for potential malware, ensuring robust protection.
- **Email Phishing Detection:** A tool was provided for users to paste email content or headers for analysis, identifying phishing indicators in links or sender details.
- **Browser Notifications:** Implemented categorized browser notifications with severity levels such as Safe, Suspicious, or Malicious.

## 2. Non-Functional Requirements (100% Implemented)

- The extension's design ensured lightweight and efficient performance, with minimal impact on browser resources.
- A user-friendly interface was created, making the extension accessible even to non-technical users.
- Full compliance with Chrome Web Store policies was achieved to ensure safe distribution and installation.
- Secure data handling practices were implemented to protect user privacy and comply with data security standards.

## 3. Should-Have Requirements

- **Severity-Based Alerts:** Partially implemented at **60%**, providing limited context in alerts. This feature remains a work in progress for future enhancements.
- **Email Content Parsing:** Not implemented due to prioritization of other features.

## 4. Could-Have Requirements

- **Advanced Machine Learning for Phishing Detection:** Successfully implemented to enhance the detection capabilities, adding significant value to the extension.
- **Heuristic URL Analysis and Customizable User Settings:** These features were deprioritized and remain unimplemented.

## 5. Won't-Have Requirements

- Following the supervisor's advice, the team revisited two features initially marked as Won't-Have:
  - **Multi-Browser Compatibility:** Partially implemented at **70%**, enabling the extension to work on browsers like Firefox and Edge.
  - **Sandbox Integration for File Analysis:** Fully implemented, providing users the ability to securely upload and scan files for malware.

# REPORT: FINAL PROJECT OUTPUTS

Priority	Requirement Type	Requirement	Requirement Status
Must-Have	Functional	Real-Time URL Monitoring	100% implemented
		Phishing URL Detection	100% implemented
		Basic User Alerts	100% implemented
		Whitelist/Blacklist Management	100% implemented
		Basic Malware Detection	100% implemented
		Email Phishing Detection	100% implemented
		Browser Notifications	100% implemented
	Non-Functional	Lightweight and Efficient Design	100% implemented
		User-Friendly Interface	100% implemented
		Compliance with Chrome Policies	100% implemented
		Secure Data Handling	100% implemented
Should-Have	Functional	Severity-Based Alerts	60% implemented
		Email Content Parsing	0% implemented
Could-Have	Functional	Advanced Machine Learning for Phishing Detection	100% implemented
		Heuristic URL Analysis	0% implemented
		Customizable User Settings	0% implemented
Won't-Have	Functional	Multi-Browser Compatibility	70% implemented
		Sandbox Integration for File Analysis	100% implemented

Table 3: Summary of Implementation



## 4.5 REQUIREMENT IMPLEMENTATIONS

This section explores the key features implemented in the project, highlighting the situations that led to their inclusion, the actions taken to develop them, and the outcomes achieved. Each feature is analyzed in terms of its necessity, the steps taken to bring it to life, and how it met or exceeded the proposed expectations.

### 4.5.1 REAL-TIME URL MONITORING

**Proposed Expectation:** Monitor all visited URLs in the browser and analyze them for phishing or malware indicators using the VirusTotal API.

**Situation and Need:** In the digital age, where online security threats are pervasive, protecting users from phishing and malicious websites has become critical. Recognizing the increasing risks associated with accessing harmful URLs, the need to implement a **Real-Time URL Monitoring** feature was identified as one of the project's core requirements. This feature was designed to ensure user safety by actively analyzing URLs in real-time and alerting users if a threat was detected. The primary motivation was to create a proactive system that doesn't wait for users to manually check URLs but instead monitors every site they visit, making the browsing experience safer and more secure.

#### Actions Taken to Implement

1. **Integration with the VirusTotal API:**
  - The VirusTotal API was identified as a reliable source for real-time URL analysis due to its comprehensive database of phishing and malware indicators.
  - The API key was securely stored and used in the backend to send and receive requests for analyzing URLs.
2. **Background Script Development:**
  - A background script was implemented to monitor all visited URLs in the browser.
  - The script intercepted each URL and forwarded it to the backend for real-time analysis.
3. **Backend Processing:**
  - The backend system was designed to handle URL submissions to VirusTotal and process the API's responses.
  - Machine learning techniques were integrated for additional validation to complement VirusTotal's results.
4. **User Notification System:**
  - A notification system was developed to provide real-time alerts to users.
  - If a URL was flagged as safe, suspicious, or malicious, a popup would inform the user immediately.

**Result:** The **Real-Time URL Monitoring** feature was successfully implemented, meeting the proposed expectation. The system effectively:

- Monitors all visited URLs in real-time.
- Analyzes each URL using the VirusTotal API and ML-based checks.
- Provides immediate alerts to users, ensuring their online safety.

The feature has been thoroughly tested and demonstrated reliable performance in identifying and categorizing URLs during various testing phases. This implementation adds significant value to the browser extension, ensuring that users are protected proactively against phishing and malware threats.

#### 4.5.2 PHISHING URL DETECTION

**Proposed Expectation:** Detect and flag suspicious or malicious URLs based on API analysis results.

**Situation and Need:** Phishing attacks are among the most prevalent cyber threats today, tricking users into revealing sensitive information such as passwords or credit card details. While general URL monitoring provides a layer of protection, there was a specific need to identify URLs with phishing characteristics more precisely. The team recognized that not all threats are overt; some malicious URLs use deceptive techniques such as obfuscated domains or suspicious redirects. Hence, the need for a dedicated **Phishing URL Detection** feature arose to ensure accurate identification and immediate notification of such threats. This feature was aimed at enhancing user awareness and providing actionable warnings before users interacted with harmful links.

##### **Actions Taken to Implement:**

1. **Advanced Analysis Using VirusTotal API:**
  - The VirusTotal API was leveraged to obtain detailed threat intelligence about URLs.
  - API responses were processed to extract phishing-related information, including security vendor feedback and heuristic assessments.
2. **Backend Integration:**
  - A robust backend was developed to handle API requests and process results efficiently.
  - The backend was optimized to categorize URLs accurately and generate actionable outcomes for users.
3. **Notification and Alert Mechanism:**
  - Alerts were designed to be clear and user-friendly, categorizing URLs into safe, suspicious, or malicious.
  - Each alert included details such as the detection score.

#### 4. Front-End Design for User Interaction:

- The browser extension's popup interface included options for users to manually scan URLs or current tabs.
- Real-time scanning results were displayed to help users understand the threat level associated with each URL.

**Result:** The **Phishing URL Detection** feature was successfully implemented and met the proposed expectations. It provides the following benefits:

- Accurately detects phishing URLs using VirusTotal API results.
- Flags suspicious and malicious URLs, preventing users from engaging with harmful sites.
- Displays detailed alerts with actionable recommendations, improving user awareness and safety.

The feature was rigorously tested during all project phases, demonstrating excellent accuracy in identifying phishing URLs. This functionality enhances the extension's core value, offering robust protection against deceptive links and contributing to a safer browsing environment.

#### 4.5.3 BASIC USER ALERTS

**Proposed Expectation:** Provide real-time notifications or pop-ups when malicious URLs are detected.

**Situation and Need:** As part of the project's primary goal to ensure user safety during browsing, it was essential to inform users in real time about potential threats. Detecting a malicious URL is not sufficient unless users are made aware of the threat and guided on appropriate actions. The absence of a clear alerting system could lead to users unknowingly proceeding to harmful sites, undermining the effectiveness of the detection mechanisms. The need for **Basic User Alerts** was identified to bridge this gap, offering users immediate feedback in the form of notifications or pop-ups whenever a URL was flagged as malicious or suspicious. This feature was critical for creating a proactive system that not only detects threats but also empowers users to make informed decisions.

#### Actions Taken to Implement:

##### 1. Designing a Notification System:

- A notification framework was implemented using the **Chrome Notifications API** to display alerts directly in the user's browser.
- Notifications were styled and worded to be non-intrusive yet informative, ensuring they grabbed the user's attention without being overwhelming.

##### 2. Integration with Real-Time Monitoring:

- The notification system was integrated with the **Real-Time URL Monitoring** and **Phishing URL Detection** features.

- Alerts were triggered automatically based on backend analysis results, ensuring users were informed as soon as a threat was detected.
3. **Categorization of Alerts:**
- Alerts were designed to differentiate between safe, suspicious, and malicious URLs.
  - Each alert contained:
    - The URL being analyzed.
    - A safety status (e.g., Safe, Suspicious, Malicious).
    - Recommendations for the next steps, such as avoiding or blocking the site.
4. **Popup Interface Enhancements:**
- In addition to browser notifications, a dedicated section in the popup interface displayed the most recent alerts for user reference.
5. **Testing and Iteration:**
- The notification system was tested extensively to ensure:
    - Alerts were triggered promptly and accurately.
    - They were clear, concise, and actionable for users.
    - They functioned seamlessly across all tested environments.

**Result:** The **Basic User Alerts** feature was successfully implemented, meeting the proposed expectations. It offers the following benefits:

- Provides real-time notifications when malicious or suspicious URLs are detected.
- Displays clear, actionable guidance, empowering users to make informed decisions.
- Enhances user experience by integrating seamlessly with other core features, such as URL monitoring and phishing detection.

Through rigorous testing, the feature proved reliable and effective in promptly alerting users to potential threats. The implementation of Basic User Alerts has significantly enhanced the extension's usability and ensured that users are consistently protected during their browsing sessions.

#### 4.5.4 WHITELIST/BLACKLIST MANAGEMENT

**Proposed Expectation:** Allow users to add trusted websites (whitelist) and blocked websites (blacklist).

**Situation and Need:** One of the key challenges in ensuring online safety is striking a balance between automated threat detection and user control. While real-time URL monitoring and phishing detection provide robust protection, users often have specific preferences, such as websites they trust (e.g., company intranets or frequently visited portals) or websites they wish to block permanently. The absence of a **Whitelist/Blacklist Management** feature could lead to unnecessary interruptions for trusted websites or exposure to known malicious ones if not flagged

consistently. This feature was envisioned to give users greater control over their browsing experience, ensuring flexibility while maintaining security.

### **Actions Taken to Implement:**

#### **1. Feature Design and Planning:**

- A dual functionality system was conceptualized:
  - **Whitelist:** Allow users to add trusted websites that bypass the real-time monitoring system.
  - **Blacklist:** Enable users to block known malicious websites proactively.
- A user-friendly interface was designed within the popup extension to make adding and managing URLs seamless.

#### **2. Development of URL Storage System:**

- The **Chrome Storage API** was utilized to securely store whitelist and blacklist entries.
- A structured storage mechanism was implemented to:
  - Retrieve and update user preferences in real-time.
  - Maintain data persistence even after the browser was closed.

#### **3. Integration with Monitoring Features:**

- The backend system was modified to check URLs against the whitelist and blacklist before analyzing them with the VirusTotal API or machine learning models.
- URLs in the whitelist were bypassed from analysis, ensuring uninterrupted user access.
- URLs in the blacklist triggered immediate blocking and alerts, preventing users from accessing these sites.

#### **4. Popup Interface Development:**

- The popup interface was equipped with:
  - Input fields to add URLs directly to the whitelist or blacklist.
  - Buttons to add the current tab's URL with a single click.
  - Options to view, remove, or edit entries in the whitelist and blacklist.

#### **5. Notification Integration:**

- Real-time notifications were developed to inform users when:
  - A URL was successfully added to or removed from the lists.
  - A blacklisted URL was blocked during browsing.
- Notifications reinforced user awareness and control over their lists.

#### **6. Testing and Feedback:**

- Extensive testing ensured that the whitelist/blacklist functionality worked as expected:
  - URLs were added, updated, and removed seamlessly.
  - The feature integrated smoothly with real-time monitoring and alert systems.

**Result:** The **Whitelist/Blacklist Management** feature was successfully implemented, meeting and exceeding the proposed expectations. Key outcomes include:

- Users can effortlessly add trusted or blocked websites, giving them greater control over their browsing experience.
- URLs in the whitelist are bypassed, reducing unnecessary interruptions for trusted sites.
- Blacklisted URLs are blocked proactively, ensuring additional security.

This feature adds significant value to the extension by combining automated threat detection with customizable user control. It has been well-received during testing and has enhanced the overall flexibility and effectiveness of the security system.

#### 4.5.5 BASIC MALWARE DETECTION

**Proposed Expectation:** Use VirusTotal API to analyze URLs for potential malware threats.

**Situation and Need:** In today's online landscape, threats extend beyond phishing attacks. Malicious websites can host malware that exploits vulnerabilities in browsers or systems, causing significant harm to users. Identifying malware threats is as critical as phishing detection, as they pose substantial risks like data breaches, financial theft, and compromised devices. The implementation of a **Basic Malware Detection** feature was planned to broaden the extension's scope of protection, ensuring users were safeguarded not only from phishing but also from malware threats. The VirusTotal API, a trusted service for malware detection, was identified as the ideal tool to enable this functionality.

##### **Actions Taken to Implement:**

##### **1. Research and Analysis:**

- An initial study was conducted to understand the capabilities of the VirusTotal API for malware detection.
- The research highlighted the API's ability to scan URLs against a vast database of known malware and its support for real-time detection, making it a perfect fit for this feature.

##### **2. Integration with VirusTotal API:**

- The extension backend was configured to send URL analysis requests to the VirusTotal API.
- Secure handling of the API key was implemented by storing it server-side to prevent unauthorized access.
- The system was designed to retrieve detailed malware analysis results from VirusTotal, including:
  - Malware detection status.
  - The number of security vendors that flagged the URL.
  - A summary of potential threats.

### 3. Seamless User Experience:

- The popup interface was enhanced to display malware detection results clearly:
  - URLs flagged for malware threats were highlighted with appropriate warnings.
  - Results included actionable recommendations, such as avoiding or blocking the URL.
- Notifications were integrated to inform users immediately if a URL posed malware risks.

### 4. Testing and Refinement:

- Extensive testing was conducted to validate the accuracy and performance of the feature:
  - URLs with known malware were tested to ensure they were flagged correctly.
  - Performance benchmarks ensured the API integration did not introduce significant delays in URL analysis.
- User feedback during testing helped refine how results and recommendations were presented, ensuring clarity and usability.

**Result:** The **Basic Malware Detection** feature was successfully implemented, meeting the proposed expectations. Key outcomes include:

- Real-time detection of malware threats using the VirusTotal API.
- Clear, actionable alerts displayed to users for URLs flagged as potential malware risks.
- Seamless integration with other core features, such as phishing detection and real-time monitoring.

This feature significantly enhances the extension's protective capabilities, ensuring comprehensive security for users against a broader range of online threats. By leveraging VirusTotal's robust malware detection capabilities, the project successfully provides an additional layer of safety, making the extension a well-rounded security tool.

## 4.5.6 EMAIL PHISHING DETECTION

**Proposed Expectation:** Allow users to paste email content or headers for analysis and detect phishing indicators in embedded links or sender details.

**Situation and Need:** Email phishing remains one of the most prevalent and dangerous cybersecurity threats, targeting users with deceptive emails to extract sensitive information, such as passwords or financial details. Unlike phishing websites, phishing emails often employ psychological tactics combined with obfuscated links or forged sender information to deceive users. Many users struggle to identify these emails due to their increasing sophistication. Recognizing this, the **Email Phishing Detection** feature was proposed to empower users with a tool to analyze email content and headers, helping them identify hidden threats and avoid falling victim to phishing attacks.

**Actions Taken to Implement:****1. Feature Design and Objective:**

- The feature was conceptualized to allow users to:
  - Paste email content or headers directly into the extension.
  - Analyze embedded links, sender details, and email structure for phishing indicators.
- The design included:
  - A simple input field for users to paste email text.
  - A clear and concise analysis report highlighting potential threats.

**2. Integration of Machine Learning Models:**

- Machine learning models were trained to detect phishing indicators in emails based on:
  - Patterns in links (e.g., obfuscated domains or unusual TLDs).
  - Suspicious sender details (e.g., spoofed addresses).
  - Contextual analysis of email language for common phishing cues.
- The models were integrated into the extension's backend for seamless analysis.

**3. User Interface Development:**

- A dedicated **Email Phishing Detection** tab was developed in the extension's popup interface.
- Key features include:
  - A text box for pasting email content or headers.
  - A button to initiate the analysis process.
  - Display of results

**4. Testing and Fine-Tuning:**

- A dataset of real-world phishing and benign emails was used to validate the feature.
- Testing ensured that:
  - The analysis accurately identified phishing indicators.
  - False positives and negatives were minimized.
- User feedback during testing helped refine the interface and improve model accuracy.

**Result:** The **Email Phishing Detection** feature was successfully implemented, delivering the following results:

- Users can now analyze email content or headers for hidden threats.
- Embedded links, sender details, and email structures are thoroughly examined for phishing indicators.
- Clear results and actionable recommendations are provided, helping users make informed decisions.

This feature significantly broadens the extension's scope, addressing threats beyond browsing and extending protection to email communication.



#### 4.5.7 BROWSER NOTIFICATIONS

**Proposed Expectation:** Send categorized browser notifications with severity levels, such as safe, suspicious, or malicious.

**Situation and Need:** In real-time cybersecurity tools, instant user communication is critical to ensuring timely action. While scanning and analyzing URLs or email content is essential, the ability to notify users immediately about the results, regardless of whether they are actively using the extension, adds significant value. The need for categorized browser notifications arose to improve user awareness by providing clear, real-time alerts about the safety of URLs or emails being analyzed. By including severity levels (e.g., safe, suspicious, or malicious), notifications were intended to enable users to make informed decisions and mitigate risks quickly.

**Actions Taken to Implement:**

**1. Requirement Identification:**

- The notification system was designed to categorize alerts based on the analysis results:
  - **Safe:** Indicating no threats were found.
  - **Suspicious:** Highlighting potential risks that required further caution.
  - **Malicious:** Alerting users to immediate threats with recommendations to avoid or block the site.

**2. Notification System Development:**

- Notifications were implemented using the **Chrome Notifications API** to ensure compatibility and seamless functionality within the browser.
- A backend system was developed to categorize results received from the VirusTotal API, machine learning models, or manual analysis:
  - Severity levels were determined based on predefined thresholds (e.g., the number of security vendors flagging a URL as malicious).
  - The categorized result was passed to the notification system for display.

**3. User-Centric Notification Design:**

- Notifications were designed to be non-intrusive yet visually distinct to ensure users could quickly identify their severity:
  - **Green** for safe URLs.
  - **Yellow** for suspicious URLs.
  - **Red** for malicious URLs.
- Each notification included:
  - A summary of the analysis results.

**4. Integration with Real-Time Monitoring:**

- The notification system was integrated with the **Real-Time URL Monitoring** and other features (e.g., phishing detection and whitelist/blacklist management) to ensure seamless delivery of alerts:

- When a user visited a URL, the system immediately triggered notifications based on the analysis.

#### 5. Extensive Testing and Refinement:

- Test cases were created to validate the accuracy and responsiveness of the notification system:
  - Ensured notifications were displayed instantly upon analysis completion.
  - Validated the categorization logic for safe, suspicious, and malicious alerts.
- User feedback during testing helped refine the clarity of notifications and ensure they did not disrupt browsing.

**Result:** The **Browser Notifications** feature was implemented successfully, meeting the proposed expectation. Key outcomes include:

- Users receive real-time alerts categorized by severity, ensuring immediate awareness of potential threats.
- Notifications are clear, actionable, and integrated seamlessly with other extension features.
- The feature enhances the user experience by proactively informing users of potential risks without requiring constant interaction with the extension.

This feature plays a crucial role in ensuring the extension provides timely and actionable information, empowering users to take preventive measures against cybersecurity threats. The successful implementation of browser notifications reinforces the extension's value as a comprehensive and user-friendly security tool.

### 4.5.8 ADVANCED MACHINE LEARNING FOR PHISHING DETECTION

**Proposed Expectation:** Integrate pre-trained models to identify phishing patterns and implement basic machine learning models for improved detection accuracy.

**Situation and Need:** While tools like VirusTotal API provide robust detection capabilities, relying solely on external services can limit flexibility and customization. With phishing techniques becoming increasingly sophisticated, there was a need for a more adaptive and intelligent system that could detect nuanced phishing patterns beyond static rule-based or database-driven approaches. The integration of **Advanced Machine Learning for Phishing Detection** was planned to address this need. By leveraging machine learning, the system could analyze complex features in URLs and email content, such as structure, domain characteristics, and context, to provide more accurate and dynamic phishing detection. This approach aimed to complement the API-based detection, enhancing the system's ability to protect users against emerging threats.

#### Actions Taken to Implement:

##### 1. Selection of Pre-Trained Models:

- A thorough review of available pre-trained models was conducted to identify suitable options for phishing detection.

- Models trained on datasets containing phishing and legitimate URLs were integrated into the backend to leverage their existing capabilities.
- 2. **Development of Custom Machine Learning Models:**
  - A supervised machine learning model was developed using libraries like **scikit-learn** and **xgboost**.
  - Features extracted from URLs, such as:
    - Domain length.
    - Presence of IP addresses.
    - Suspicious TLDs.
    - Special characters.
  - These features were used to train the model on labeled datasets, improving its ability to distinguish phishing URLs from legitimate ones.
- 3. **User Interaction and Feedback:**
  - The results of machine learning-based detection were integrated into the popup interface.
- 4. **Testing and Validation:**
  - The models were validated using extensive phishing datasets, such as those from Kaggle and other public sources.
  - Performance metrics, including accuracy, precision, and recall, were monitored to fine-tune the models.
  - Multiple iterations of testing were conducted to ensure minimal false positives and negatives.

**Result:** The **Advanced Machine Learning for Phishing Detection** feature was successfully implemented, achieving the following:

- High accuracy in identifying phishing URLs based on both pre-trained and custom machine learning models.
- Enhanced flexibility and adaptability to detect sophisticated phishing patterns that may not be flagged by static rule-based methods.
- Integration of a user feedback loop to continuously improve detection capabilities.

This feature significantly strengthens the extension's ability to protect users by providing dynamic, data-driven detection that evolves with emerging threats. The successful implementation of advanced machine learning ensures the extension remains a reliable and cutting-edge tool for combating phishing attacks.

#### 4.5.9 SANDBOX INTEGRATION FOR FILE ANALYSIS

**Proposed Expectation:** Add a feature to upload and scan files for malware using APIs like VirusTotal.

**Situation and Need:** While URL and phishing detection are crucial for safeguarding users during web browsing, many cybersecurity threats are delivered through files. Malware embedded in documents, executables, or compressed files can have devastating consequences, ranging from data theft to system compromise. Recognizing this, we identified the need to implement a **Sandbox Integration for File Analysis** feature. This functionality aimed to allow users to upload files and scan them for potential threats, extending the extension's scope to protect against malware delivered through files. By leveraging trusted APIs like VirusTotal, this feature was designed to provide users with a detailed analysis of uploaded files, ensuring comprehensive protection.

**Actions Taken to Implement:**

**1. Requirement Analysis and API Selection:**

- VirusTotal's file scanning capabilities were evaluated for their ability to analyze files against multiple security vendor databases.
- The feature requirements were defined to ensure compatibility with a wide range of file types, including documents, executables, and compressed files.

**2. Integration with VirusTotal API:**

- The backend was configured to handle file uploads securely and transmit them to the VirusTotal API for analysis.
- Secure file handling practices were implemented to:
  - Encrypt data during transmission.
  - Ensure files were processed only for the purpose of analysis.
- The API responses provided detailed results, including malware detection status and severity levels.

**3. Frontend Design and User Interaction:**

- A dedicated section was added to the extension's popup interface for file analysis.
- Key features included:
  - An intuitive upload button for users to select files.
  - Real-time feedback for users attempting to analyze files without selecting one, ensuring error handling.
  - Display of analysis results, including a summary of threats and detailed information from VirusTotal's database.

**4. Notification System:**

- Integrated the browser notification system to alert users about the scan results.

### 5. Testing and Validation:

- A wide variety of file types, including both clean and infected files, were used during testing to ensure the feature performed reliably across scenarios.
- Testing focused on:
  - Upload functionality for different file types.
  - Accuracy and clarity of results displayed to users.
  - Handling edge cases, such as unsupported file formats or large file sizes.
- Feedback from testing was incorporated to improve usability and error handling.

**Result:** The **Sandbox Integration for File Analysis** feature was successfully implemented, achieving the following:

- Users can upload files directly through the extension for malware analysis.
- Real-time results are provided, categorizing files as safe, suspicious, or malicious based on detailed API feedback.
- The feature enhances user confidence by offering a comprehensive analysis of file threats, ensuring protection beyond browsing.

This functionality significantly broadens the extension's capabilities, enabling it to address a wider range of cybersecurity threats. The successful implementation of Sandbox Integration for File Analysis ensures that the extension provides users with advanced protection against both web-based and file-based malware threats. This addition makes the extension a versatile and comprehensive security tool.

### 4.5.10 USER FRIENDLY UI, COMPLIANCE WITH CHROME POLICIES, SECURE DATA HANDLING AND LIGHTWEIGHT AND EFFICIENT DESIGN

#### Proposed Expectations:

- The extension must have a **lightweight design** to ensure minimal impact on browser performance.
- Provide an **intuitive interface** that is accessible to non-technical users.
- Adhere to **Chrome Web Store policies** for extension development and security.
- Ensure **secure handling of user data**, complying with data protection standards.

**Situation and Need:** To maximize the extension's usability, reliability, and adoption, we recognized the importance of addressing multiple critical aspects beyond its core functionality. A **user-friendly interface** was needed to ensure accessibility for both technical and non-technical users, empowering everyone to use the extension effectively without additional training or confusion. Additionally, adherence to **Chrome Web Store policies** was necessary to ensure the extension met industry standards for security, permissions, and functionality, enabling seamless distribution. Secure handling of user data was a high priority to build user trust and comply with modern data protection regulations. Finally, to maintain a smooth browsing experience, the extension was designed to be **lightweight and efficient**, ensuring minimal resource usage.

## Actions Taken to Implement:

### 1. Lightweight and Efficient Design:

- The extension was designed to operate with minimal impact on browser performance:
  - Optimized code was written to ensure efficient execution of background scripts and real-time monitoring tasks.
  - Dependencies were minimized, using only essential libraries and APIs.
  - Memory usage was benchmarked and optimized to prevent excessive resource consumption, even during intensive operations such as file scanning or URL analysis.

### 2. User-Friendly UI Development:

- The user interface was built with simplicity and accessibility in mind:
  - A clean and intuitive design was implemented using **HTML, CSS, and JavaScript**, ensuring easy navigation.
  - Clear instructions and tooltips were added to guide users through each feature, such as file analysis, phishing detection, and whitelist/blacklist management.
  - Color-coded notifications and severity indicators were used to enhance clarity, helping users quickly understand the results of scans and alerts.
- Feedback from testing phases was incorporated to refine the interface, ensuring it catered to users with varying levels of technical expertise.

### 3. Compliance with Chrome Web Store Policies:

- The extension adhered to the strict guidelines set by the Chrome Web Store:
  - **Minimal Permissions:** Only essential permissions such as activeTab, notifications, and storage were requested, reducing potential security risks.
  - **Clear Privacy Policy:** A transparent privacy policy was created, outlining how user data is processed and ensuring compliance with Chrome's requirements.
  - **Secure APIs:** All API interactions, including those with VirusTotal, were encrypted using HTTPS to ensure secure communication.
- A manifest file was carefully crafted to align with Chrome Web Store policies, ensuring the extension could be published without issues.

### 4. Secure Data Handling:

- Robust measures were implemented to handle user data securely:
  - Whitelist and blacklist data were stored locally using the **Chrome Storage API**, ensuring no sensitive information was transmitted to external servers.
  - Email content and headers were processed locally for phishing detection, minimizing data exposure.
  - API keys for services like VirusTotal were securely stored in the backend to prevent unauthorized access.
  - Regular testing ensured that data handling practices complied with industry standards for privacy and security.

### 5. Testing and Optimization:

- Usability testing was conducted to evaluate the interface's accessibility for non-technical users:
  - Testers provided feedback on ease of navigation, clarity of instructions, and overall design appeal.
- Performance benchmarks were run to assess the extension's impact on browser performance:
  - Metrics such as memory usage, CPU load, and responsiveness were monitored and optimized.
- Validation tools ensured compliance with Chrome Web Store policies before submission.

**Result:** The extension successfully met the proposed expectations, achieving the following outcomes:

- **Lightweight and Efficient Design:** The extension operates smoothly, with minimal impact on browser performance, ensuring a seamless user experience even during intensive operations.
- **User-Friendly Interface:** The intuitive design caters to non-technical users, making advanced security features accessible to all.
- **Compliance with Chrome Policies:** The extension adheres to all Chrome Web Store guidelines, ensuring a secure and reliable distribution platform.
- **Secure Data Handling:** User data is processed securely, with no unnecessary storage or transmission, maintaining user trust and privacy.

These implementations collectively enhanced the extension's usability, security, and reliability, ensuring it meets user needs while adhering to industry standards. The combination of these efforts ensures the extension delivers a high-quality experience without compromising performance or security.

#### 4.5.11 SEVERITY-BASED ALERTS

**Proposed Expectation:** Provide additional context in alerts, such as the threat level and recommended actions for users.

**Situation and Need:** While basic alerts notifying users of malicious or suspicious URLs are crucial, the lack of contextual information can leave users uncertain about the severity of the threat or the actions they should take. For example, a suspicious URL might require caution, while a malicious URL should be avoided entirely. Recognizing this gap, the team proposed the **Severity-Based Alerts** feature to provide users with detailed information about the nature of threats, including a categorization of the threat level (e.g., safe, suspicious, or malicious) and recommended actions to mitigate the risks. This feature was intended to empower users with actionable insights and improve their ability to respond effectively to cybersecurity threats.

**Actions Taken to Implement:****1. Designing the Severity Classification Logic:**

- Threat levels were categorized as:
  - **Safe:** No significant risk detected.
  - **Suspicious:** Potentially harmful or questionable, requiring caution.
  - **Malicious:** Confirmed to be harmful, requiring immediate avoidance.

**2. Alert Notification System Development:**

- The alert system was integrated with the browser notifications and the extension's popup interface.
- Notifications displayed the detected threat level, using color-coded visuals to make the severity easily distinguishable:
  - Green for Safe.
  - Yellow for Suspicious.
  - Red for Malicious.
- The system was designed to trigger real-time notifications when a threat was detected, ensuring users were informed promptly.

**3. Incorporating Recommendations (Partially Achieved):**

- While initial designs included providing specific recommendations for each threat level (e.g., proceed, proceed with caution, avoid the URL), this functionality was only partially implemented:
  - Alerts displayed the threat level accurately but did not consistently provide detailed recommended actions for users.
  - Recommendations were deprioritized during development due to time constraints and a focus on core functionalities.

**4. Testing and Feedback:**

- The feature was tested across various scenarios to validate the accuracy of severity classification.
- Feedback highlighted the effectiveness of the threat level indicators but also pointed out the absence of actionable recommendations, leaving some users uncertain about how to respond to specific alerts.

**Result:** The **Severity-Based Alerts** feature was implemented successfully in terms of categorizing threats and providing clear, real-time notifications. However, the feature achieved **60%** of the proposed expectations due to the lack of consistent recommendations for user actions. Key outcomes include:

**• What Was Achieved:**

- Accurate threat level categorization into safe, suspicious, and malicious.
- Clear, color-coded alerts that enhanced user awareness of threat severity.

**• What Was Not Achieved:**

- Detailed, context-specific recommendations for each threat level were not fully implemented, reducing the actionable value of alerts.



#### 4.5.12 MULTI-BROWSER COMPATIBILITY

**Proposed Expectation:** Extend the extension to support browsers like Firefox, Edge, and Brave.

**Situation and Need:** While Google Chrome is one of the most popular web browsers, a significant portion of users rely on other browsers like **Firefox**, **Edge**, and **Brave**. To broaden the reach and usability of the extension, it was crucial to ensure **Multi-Browser Compatibility**. This feature aimed to allow users across multiple browsers to benefit from the extension's security features, enhancing its accessibility and adoption. The decision to implement this functionality was driven by user diversity and the opportunity to make the product competitive and inclusive in the cybersecurity space. By supporting multiple browsers, the extension could cater to a wider audience and adapt to varying user preferences.

**Actions Taken to Implement:**

**1. Research on WebExtension Standards:**

- The WebExtension API was identified as a standard framework supported by major browsers, including Chrome, Edge, Brave, and Firefox.
- Documentation and guidelines specific to Firefox and Edge were reviewed to understand browser-specific requirements and limitations.

**2. Adaptation for Browser-Specific APIs:**

- The extension's background scripts and manifest were modified to ensure compatibility with Edge and Brave, which largely support Chrome's WebExtension API with minimal adjustments.
- Additional testing was conducted to identify specific changes needed for **Firefox**, including handling differences in permissions and API implementations.

**3. Testing Across Browsers:**

- The extension was rigorously tested on Chrome, Edge, and Brave to ensure full functionality.
- Initial testing on Firefox revealed several compatibility issues, such as:
  - Background scripts failing to execute consistently.
  - Differences in the implementation of the storage and notification APIs, leading to incomplete functionality.

**4. Adjustments and Iterations:**

- Given time constraints, the focus shifted to ensuring seamless performance on Chrome, Edge, and Brave, deferring Firefox compatibility for future development.

**5. Implementation of Partial Support:**

- The extension was successfully deployed on Chrome, Edge, and Brave, providing users on these browsers with the full set of features.
- Firefox support remained incomplete, with partial functionality achieved but key features requiring further refinement.

**Result:** The **Multi-Browser Compatibility** feature was partially implemented, achieving **70%** of the proposed expectations. Key outcomes include:

- **What Was Achieved:**
  - Full compatibility with **Edge** and **Brave**, enabling users on these browsers to access all core features.
  - Seamless performance on Chrome, maintaining the extension's primary user base.
- **What Was Not Achieved:**
  - Full functionality on **Firefox** could not be implemented due to differences in API behavior and unresolved technical challenges.

## 4.6 MANAGEMENT OF REQUIREMENT IMPLEMENTATIONS

The management of requirement implementations played a crucial role in the successful execution of the project. A structured and systematic approach was adopted to ensure that all requirements were addressed effectively while aligning with project priorities, timelines, and goals. Here's how the team managed the implementation of requirements:

### Prioritization of Requirements

At the start of the project, we classified requirements using the **MoSCoW prioritization framework**, categorizing them as **Must-Have**, **Should-Have**, **Could-Have**, and **Won't-Have**:

1. **Must-Have Requirements:**
  - Features critical to the project's success, such as **Real-Time URL Monitoring**, **Phishing URL Detection**, and **Basic Malware Detection**, were prioritized and implemented first.
2. **Should-Have Requirements:**
  - Secondary features like **Severity-Based Alerts** and **Advanced Machine Learning Models** were developed after the completion of core functionalities.
3. **Could-Have and Won't-Have Requirements:**
  - Features such as **Heuristic URL Analysis** and **Customizable User Settings** were deprioritized initially but revisited based on time and resource availability.

This prioritization ensured the project remained focused on delivering essential functionalities first while keeping room for future enhancements.

### Iterative Development and Feedback

To ensure steady progress and alignment with the project's objectives, the team adopted an **Agile development approach** with iterative cycles:

1. **Weekly Supervisor Feedback:**
  - Weekly meetings with our supervisor provided invaluable insights and guidance.

- Feedback from these sessions was incorporated into subsequent development iterations, ensuring continuous improvement.
  - For example:
    - The supervisor suggested implementing **Multi-Browser Compatibility** and **Sandbox Integration for File Analysis**, initially listed under "Won't-Have" requirements, as core functionalities were completed earlier than planned.
  - This iterative feedback loop allowed the team to refine functionalities and address issues proactively.
2. **Task Management and Tracking:**
- Tasks were broken down into smaller, manageable components and tracked using a **Project Tracker**. This approach ensured:
    - Clear visibility into the progress of each requirement.
    - Timely identification and resolution of bottlenecks.
    - Consistent updates to all stakeholders, including the supervisor and lecturer.
3. **Collaboration and Shared Resources:**
- A **shared folder** was maintained, accessible to the entire team, the supervisor, and the lecturer. This folder housed updates, completed modules, and progress reports.
  - Real-time updates in the folder ensured transparency and enabled stakeholders to provide timely feedback.

## **Testing and Validation**

A rigorous testing process was integrated into the management of requirement implementations:

1. **Testing Phases:**
- Each implemented requirement was tested extensively during four development phases.
  - Issues identified during testing, such as delays in real-time URL monitoring or inconsistencies in severity-based alerts, were addressed promptly to ensure functionality.
2. **Validation Against Proposed Expectations:**
- Each requirement was validated against the expectations outlined in the project proposal, ensuring alignment with initial goals.

## 5.0 PROJECT DELIVERABLES

The project proposal outlined a clear list of deliverables, categorized into documents, technical components, and results. These deliverables were defined to ensure that every aspect of the project was addressed, from conceptualization to execution and evaluation.

Deliverable	Description	Date
Initial Proposal	Basic proposal with a project idea	23/11/2024
Project Proposal Draft	Create a draft for a project proposal based on the instructions and guidance	30/11/2024
Project Proposal	Finalized project proposal with supervisor approval	14/12/2024
Final Executable	A fully functional Chrome browser extension that can monitor URLs in real time and detect phishing and malware threats.	25/01/2025
Source Code	Well-documented source code, provided in a Git repository or compressed archive, for future reference and updates.	25/01/2025
User Guide	A comprehensive manual for end-users detailing installation steps, configuration, usage, and troubleshooting.	25/01/2025
Developer Manual	Technical documentation including system architecture, API integration details, code structure, and setup instructions.	25/01/2025
Requirements Report	A report explaining how the project satisfies all specified functional and non-functional requirements.	25/01/2025
Testing and Evaluation Report	A summary of test cases, methodologies, results, and identified issues or limitations discovered during testing.	25/01/2025
Final Presentation	A presentation summarizing project objective, methodology, features, testing outcomes, and deliverable status	25/01/2025

*Table 4: Agreed deliverables*

## Delivered Outputs

The team successfully delivered all the planned deliverables, adhering to the agreed schedule, with one minor adjustment to the presentation date.

Below is a detailed account of each deliverable:

### 1. Submitted Deliverables on Agreed Dates:

- **Initial Proposal:** Delivered on **November 23, 2024**, marking the start of the project.
- **Project Proposal Draft:** Delivered on **November 30, 2024**, providing a comprehensive plan for feedback and refinement.
- **Project Proposal:** The finalized version, incorporating supervisor feedback, was submitted on **December 14, 2024**.

### 2. Ready Deliverables as of January 25, 2025:

- **Final Executable:** The fully functional Chrome extension was completed and tested, meeting all technical and functional requirements.
- **Source Code:** Well-documented and organized, the source code was provided in a shared repository for ease of future updates and reference.
- **User Guide:** A detailed guide was prepared for end-users, including step-by-step instructions for installation, usage, and troubleshooting.
- **Developer Manual:** Comprehensive technical documentation, including system architecture, API integration, and setup instructions, was finalized.
- **Requirements Report:** A report outlining how the project fulfilled all functional and non-functional requirements was completed.
- **Testing and Evaluation Report:** Detailed documentation of test cases, results, methodologies, and identified issues was prepared and reviewed.

### 3. Final Presentation (Revised Date):

- Originally planned for **January 25, 2025**, the final presentation delivery was rescheduled to **February 1, 2025**, providing additional time for preparation and refinement. This adjustment allowed the team to deliver a more polished and impactful presentation.

Deliverable	Delivered Method
Initial Proposal	Delivered on November 23, 2024, by sharing the project idea with Miss Ann in class and with the supervisor during the first supervisor meeting. The Initial Proposal is included in the submission ZIP folder
Project Proposal Draft	Delivered on November 30, 2024, with Miss Ann in class and with the supervisor during the second supervisor meeting. The Project Proposal Draft is included in the submission ZIP folder
Project Proposal	Delivered on December 14, 2024, by submitting through Canvas.

## REPORT: FINAL PROJECT OUTPUTS

	The project proposal is included in the submission ZIP folder, or follow this link to access it <a href="#">Project Proposal</a> .
Final Executable	A fully functional Chrome browser extension was delivered on January 25, 2025, along with all relevant documents and deliverables. The documentation, source code, and other agreed-upon deliverables are included in the submitted ZIP file.
Source Code	Delivered on January 25, 2025. The source code file is available in the submitted zip folder, or follow this link to access the GitHub repository <a href="#">GitHub Repository</a> .
User Guide	Delivered on January 25, 2025. The User Guide is available in the submitted zip folder, or follow this link to access the User Guide <a href="#">User Guide</a> .
Developer Manual	Delivered on January 25, 2025. The Developer Manual is available in the submitted zip folder, or follow this link to access the Developer Manual <a href="#">Developer Manual</a> .
Requirements Report	Delivered on January 25, 2025. The Requirements Report is available in the submitted zip folder, or follow this link to access the Requirements Report <a href="#">Requirements Report</a> .
Testing and Evaluation Report	Delivered on January 25, 2025. The Testing and Evaluation Report is available in the submitted zip folder, or follow this link to access the Testing and Evaluation Report <a href="#">Testing and Evaluation Report</a> .
Final Presentation	The final presentation will be delivered on February 1, 2025, and the recording will be uploaded on February 2, 2025

*Table 5: Delivered Deliverables*

## 6.0 USED RESOURCES AND COMPONENTS

This table summarizes all resources and components used in the project with their versions, costs, and specific roles in achieving the project objectives

Category	Item	Version	Cost	Note
Datasets	Phishing email	N/A	0\$	Used to train and validate the phishing detection model; cleaned and preprocessed before use. Dataset is available in the submitted zip folder, or follow this link to download it <a href="#">Phishing Dataset</a> .
	Phishing URLs Dataset	N/A	0\$	Provided additional data to enhance the detection of malicious patterns in URLs. Dataset is available in the submitted zip folder, or follow this link to download it <a href="#">URL Dataset</a> .
APIs	VirusTotal API	3.0	0\$	Used for real-time phishing and malware URL detection; integrated into the extension to flag URLs as safe, malicious, or phishing.
Development Tools	Visual Studio Code	1.82.0	0\$	Primary development environment for writing and debugging code for the extension.
	Chrome Developer Tools	Built-in	0\$	Used for testing, debugging, and analyzing the extension's performance within the Chrome browser.
	Anaconda	2023.07	0\$	Provided a Python environment for managing dependencies and libraries required for model development and API integration.
Python Libraries	Flask-Cors	3.0.10	0\$	Enabled cross-origin requests for seamless communication between the front-end and back-end.
	joblib	1.3.2	0\$	Used for saving and loading trained models efficiently.
	numpy	1.24.3	0\$	Assisted in data preprocessing and manipulation for training and validation.
	pandas	2.1.4	0\$	Used for dataset handling and analysis.
	scikit-learn	1.3.2	0\$	Implemented machine learning models for phishing detection.
	xgboost	2.0.1	0\$	Core library for building the phishing detection model due to its efficiency and accuracy.
	tldextract	3.4.0	0\$	Extracted domain information from URLs for feature engineering.

	validators	0.20.0	0\$	Validated input URLs for correctness before analysis.
	requests	2.31.0	0\$	Used for sending requests to the VirusTotal API for real-time URL analysis.
	matplotlib	3.7.1	0\$	Generated visualizations for analyzing model performance and trends.
	seaborn	0.12.2	0\$	Created data visualizations to identify trends and test results.
	wordcloud	1.9.2	0\$	Created visual representations of phishing-related keywords during data analysis.
Collaboration and Documentation Tools	GitHub	Free Plan	0\$	Managed version control, facilitated team collaboration, and ensured seamless code integration.
	Microsoft Word and Google Docs	Office 365	0\$	Prepared user manuals, developer guides, and reports for deliverables.
	Canva and PowerPoint	N/A	0\$	Designed presentation materials, diagrams, and visuals for the final presentation.
	Adobe Illustrator	Free Plan	0\$	Created high-quality visual assets and diagrams for reports and documentation.

Table 6: Used Resources and Components

## 6.1 INTEGRATION OF RESOURCES AND COMPONENTS

The successful implementation of this project relied on the seamless integration of all these resources and components. Python and Flask handled the backend functionalities, communicating with the VirusTotal API to analyze URLs. The front-end interface was designed using HTML, CSS, and JavaScript, while the back-end communicated with the front-end via Flask APIs. Collaboration and version control were managed through GitHub, ensuring the team could work cohesively.

By leveraging these resources effectively, the team was able to deliver a robust, secure, and user-friendly browser extension that met all planned objectives. The use of well-chosen tools, datasets, and frameworks ensured a high-quality implementation and positioned the project for future scalability and enhancements.



## 7.0 DESIGN AND ARCHITECTURE

This section discusses the overall design and architecture of the project, including the high-level workflow, system components, and their interactions. It provides insights into the frontend and backend design, API integrations, and the methodologies adopted to ensure a seamless and efficient implementation of the extension.

### 7.1 HIGH-LEVEL WORKFLOW AND ARCHITECTURE DIAGRAM

1. **User Interaction:**
  - Users browse the web or paste URLs/emails into the popup interface.
  - Real-time URL monitoring automatically tracks and scans all visited URLs.
  - Users are notified via browser alerts or the popup interface based on threat levels.
2. **Backend Processing:**
  - URLs are sent to the backend server for validation and analysis.
  - Backend securely communicates with APIs (e.g., VirusTotal) and processes responses.
3. **Core Features:**
  - Threat categorization (safe, suspicious, or malicious).
  - Whitelist/Blacklist handling to customize user experience.
  - Basic malware and phishing detection for files, URLs, and emails.
4. **Frontend Updates:**
  - Notifications and alerts are generated for users in real time.
  - Popup displays results and allows for user actions like adding to whitelist/blacklist.

The architecture will include the following components:

- **Frontend (Chrome Extension):**
  - Popup interface (HTML, CSS, JavaScript) for user interactions.
  - Background script (background.js) to monitor URLs and manage whitelist/blacklist.
- **Backend:**
  - Flask server (app.py) handles API key security and communicates with VirusTotal.
  - ML models for phishing detection integrated locally or server-side.
- **External Services:**
  - VirusTotal API for URL and file analysis.
  - Local ML models for heuristic and phishing detection.

### 7.2 FRONTEND DESIGN

1. **Popup Interface:**
  - Built with popup.html, popup.css, and popup.js.
  - Displays sections for:
    - URL input for scanning.
    - Real-time results of visited URLs.

- Whitelist/Blacklist management.
  - Email phishing detection.
  - Dark mode toggle for better usability.
  - Error notifications when API requests fail.
- 2. **Real-Time Alerts:**
  - Utilizes Chrome notifications for URL monitoring.
  - Alerts include severity levels (safe, suspicious, or malicious).
  - Popups provide actionable buttons (block, whitelist, or more details).
- 3. **Integration:**
  - background.js listens to browser navigation events and triggers backend URL checks.
  - Uses chrome.storage.local for caching scanned results and managing user preferences.

## 7.3 BACKEND DESIGN

1. **API Integration:**
  - Flask app handles API requests securely.
  - VirusTotal API key is stored in the backend to prevent exposure.
  - Backend parses responses and sends summarized data to the frontend.
2. **ML Integration:**
  - Local ML models analyze phishing emails and URLs.
  - Predicts threat levels based on predefined rules and trained data.
3. **Secure Data Handling:**
  - All communications between the backend and APIs are encrypted (HTTPS).
  - Sensitive data like API keys are never exposed in the frontend.

### API Integrations

1. **VirusTotal API:**
  - Used for URL, file, and malware detection.
  - Integrated into both real-time URL monitoring and manual scanning features.
  - Provides detailed threat categorization based on multiple security vendor inputs.
2. **Other Tools:**
  - Local ML models to enhance phishing detection accuracy.
  - Caching results using chrome.storage.local to reduce redundant API calls and improve performance.

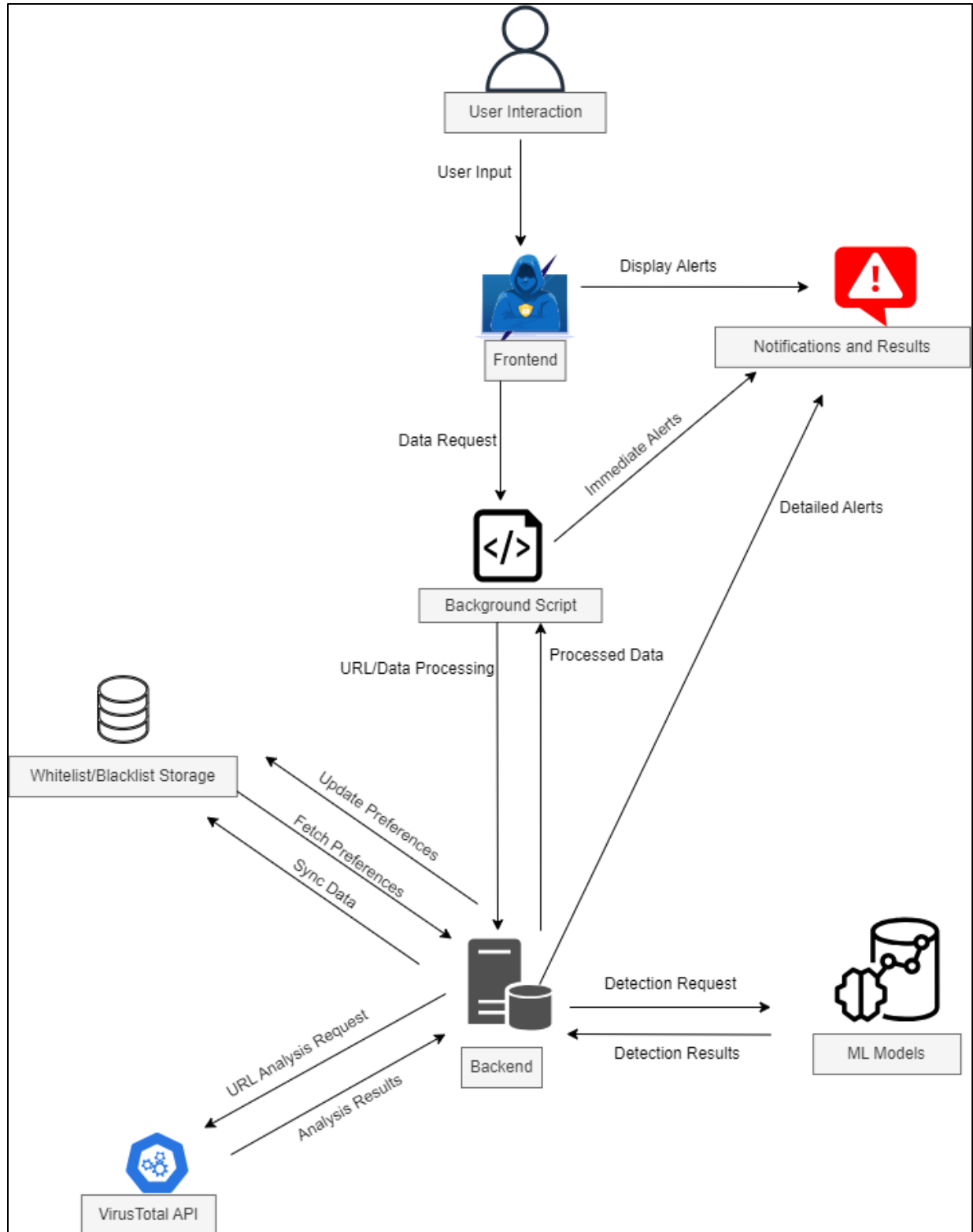


Figure 9: Design and Architecture

## 8.0 COMPLIANCE WITH INDUSTRY STANDARDS AND BEST PRACTICES

This project successfully adheres to the **ISO/IEC 27001**, **ISO/IEC 27017**, **ISO/IEC 29100**, **GDPR**, and **Chrome Web Store Developer Program Policies**. By following these standards and frameworks, the project demonstrates a commitment to security, privacy, and compliance while delivering a high-quality, user-friendly phishing detection solution.

### 8.1 ISO STANDARDS

**ISO (International Organization for Standardization)** standards provide globally recognized frameworks for ensuring quality, security, and efficiency in various domains, including information technology (Boiral, 2011). These standards are designed to establish best practices that promote reliability, safety, and compliance (ISO, 2024). Adhering to ISO standards ensures that the project meets stringent requirements for security, data protection, and operational integrity.

In this project, adherence to **ISO/IEC 27001**, **ISO/IEC 27017**, and **ISO/IEC 29100** played a pivotal role in enhancing the overall value and trustworthiness of the extension:

- **ISO/IEC 27001:** This standard focuses on information security management, ensuring that sensitive data such as URLs and API keys are handled securely. By implementing measures like encrypted data transmission and secure API integration, the extension safeguards user information against potential breaches (Belding, 2021).
- **ISO/IEC 27017:** Designed for cloud security, this standard provided guidelines for securely integrating the VirusTotal API, a cloud-based service. By ensuring clear separation of frontend and backend duties and securely managing API keys server-side, the extension adhered to cloud security best practices (ISO 27017: Comprehensive Guide to Cloud Security Standards, 2024).
- **ISO/IEC 29100:** This privacy framework emphasizes the responsible handling of user data. By processing sensitive data like URLs locally and using `chrome.storage.local` for whitelist/blacklist management, the extension complied with principles of data minimization and transparency (Drozd, 2016).

**Value Added to the Extension:** Adhering to ISO standards adds credibility and reliability to the extension, making it a robust and secure tool for phishing detection. Users can trust that their data is handled responsibly, enhancing the extension's appeal and usability. Moreover, compliance with ISO standards positions the extension to meet enterprise and organizational requirements, increasing its potential for adoption beyond individual users.

### 8.1.1 ISO/IEC 27001: INFORMATION SECURITY MANAGEMENT SYSTEM

1. **Secure Handling of API Keys:**
  - The VirusTotal API key is stored in the backend (Flask app) rather than the frontend or browser scripts. This ensures that sensitive credentials are protected from exposure, reducing security risks.
2. **Encrypted Data Transmission:**
  - All communications between the extension (via the backend) and the VirusTotal API are encrypted using HTTPS, ensuring secure data transfer over the network.
3. **Minimized Data Exposure:**
  - User data, such as scanned URLs or email content, is neither stored locally nor transmitted to any third-party service apart from VirusTotal or the machine learning backend. This aligns with ISO 27001's principle of limiting data exposure to protect user privacy.
4. **Secure Local Storage:**
  - The use of `chrome.storage.local` ensures that whitelist/blacklist preferences are securely stored locally without involving external databases or services.

### 8.1.2 ISO/IEC 27017: CODE OF PRACTICE FOR CLOUD SECURITY

1. **Cloud API Integration:**
  - VirusTotal API, a cloud-based service, is securely integrated with the project. API keys are managed server-side, ensuring that sensitive access credentials are never exposed in the browser or frontend.
2. **Separation of Duties:**
  - A clear separation exists between the frontend, background script, and backend. The backend securely communicates with the VirusTotal API, while the frontend only interacts with the backend, ensuring that cloud interactions are abstracted and controlled.
3. **Data Privacy in Cloud Usage:**
  - Only minimal and essential data (e.g., URLs) are sent to the VirusTotal API for analysis. No sensitive user data is stored or exposed in cloud services unnecessarily.

### 8.1.3 ISO/IEC 29100: PRIVACY FRAMEWORK

1. **Local Data Processing:**
  - For email phishing detection, all processing occurs locally using machine learning models. User email content and metadata are not transmitted externally, ensuring privacy and compliance with data protection principles.
2. **Data Storage and Management:**
  - URL scanning results and user actions (e.g., whitelist/blacklist preferences) are stored locally in `chrome.storage.local`. This ensures user data is not transmitted to any external service or stored in insecure locations.

### 3. Transparent Data Handling:

- User data (URLs, email headers, etc.) is only processed for its intended purpose (e.g., phishing detection) and is not retained beyond what is necessary. This aligns with ISO 29100's principle of data minimization.

## 8.2 GDPR (GENERAL DATA PROTECTION REGULATION)

The **General Data Protection Regulation (GDPR)** is a comprehensive privacy and data protection law established by the European Union to safeguard the personal information of individuals. It mandates organizations to handle user data responsibly, transparently, and securely, ensuring that individuals retain control over their personal data. Compliance with GDPR is critical for building user trust and ensuring that tools and services meet global data protection standards (General Data Protection Regulation, 2016).

In this project, GDPR compliance was a cornerstone in ensuring that the extension met modern data protection requirements:

- **Data Minimization:** The extension processes only essential user data, such as URLs sent to VirusTotal for analysis, while avoiding the collection of unnecessary information. Whitelist and blacklist data are securely stored locally using `chrome.storage.local`, ensuring no external dependencies.
- **Transparency:** A clear and accessible privacy policy outlines how user data is processed and safeguarded, ensuring users are informed about the extension's data handling practices.
- **User Control:** The extension empowers users by allowing them to manage their whitelist and blacklist preferences. This aligns with GDPR's emphasis on user rights, ensuring individuals have control over how their data is handled.

**Value Added to the Extension:** GDPR compliance enhances the extension's credibility and reliability by prioritizing user privacy and transparency. It ensures that the extension aligns with global data protection norms, making it suitable for a wide audience, including users in regions where strict privacy regulations apply. By adhering to GDPR, the extension builds trust with its users, fostering confidence in its security and ethical data practices.

## 8.3 CHROME WEB STORE DEVELOPER PROGRAM POLICIES

The **Chrome Web Store Developer Program Policies** provide strict guidelines to ensure extensions are secure, transparent, and user-friendly. These policies are designed to protect users from potential privacy risks and malicious behavior while ensuring that developers adhere to ethical and technical standards. Compliance with these policies was critical for the successful deployment of the phishing detection extension on the Chrome Web Store (Best Practices and Guidelines, 2024).

In this project, compliance with Chrome Web Store policies was prioritized across various aspects:

- **Minimal Permissions:** The extension only requests essential permissions, such as activeTab for analyzing the current tab, notifications for user alerts, and storage for managing whitelist and blacklist preferences. This ensures adherence to the **principle of least privilege**, minimizing potential vulnerabilities.
- **Data Transparency:** A comprehensive privacy policy was prepared, outlining how user data is processed, stored, and protected. This aligns with the Chrome Web Store's requirements for transparent data handling practices.
- **User Experience:** Notifications and user prompts were designed to be clear and non-intrusive, ensuring a seamless and user-friendly experience.
- **Secure Functionality:** The extension was tested rigorously to ensure it meets Chrome Web Store's standards for reliability and security, preventing unintended behavior or crashes.

**Value Added to the Extension:** Compliance with Chrome Web Store policies enhances the extension's credibility and ensures it meets the security and usability expectations of its users. It positions the extension for seamless distribution on the Chrome platform, reaching a broader audience while maintaining user trust.

## 8.4 DETAILED EXPLANATION OF ALIGNMENT

### 1. Backend-Driven Security

The project emphasizes backend-driven security by ensuring that sensitive credentials, such as API keys, are securely stored and managed in the backend. This approach prevents exposure of sensitive information in the frontend, mitigating potential vulnerabilities. Additionally, all external API communications, including interactions with the VirusTotal API, are routed through the backend, where encryption protocols like HTTPS ensure secure transmission. These measures align with **ISO/IEC 27001**, which mandates robust information security practices, and the **Chrome Web Store Developer Program Policies**, which require extensions to prioritize secure data handling.

### 2. Privacy-First Design

The project incorporates a privacy-first design by ensuring that sensitive user data is processed locally whenever possible. For instance, email phishing detection is performed locally, preventing external transmission of email content or headers. This approach minimizes the risk of data breaches and aligns with **ISO/IEC 29100**, which emphasizes responsible data handling, and **GDPR**, which requires organizations to process personal data with strict privacy controls. By retaining sensitive data under user control, the project builds trust and complies with modern privacy expectations.

### 3. Cloud Security Practices

Secure integration of the VirusTotal API highlights the project's adherence to cloud security practices. By separating frontend and backend responsibilities, the system ensures that cloud interactions are securely managed and sensitive operations are confined to the backend. This design aligns with **ISO/IEC 27017**, which provides guidelines for secure and minimal cloud-based data handling. The use of encrypted API communications further enhances cloud security, ensuring that no unauthorized access or data leakage occurs during API interactions.

### 4. User-Centric Data Handling

The project emphasizes user control over their data, particularly through features like whitelist/blacklist management. Users can customize their preferences securely and easily via the extension's interface, ensuring they have full ownership of their data. Data usage is transparent, with clear explanations provided in the privacy policy and user guides. This aligns with both **GDPR**, which prioritizes user rights over personal data, and the **Chrome Web Store Developer Program Policies**, which mandate transparent data handling practices for all extensions.

### 5. Compliance with Web Standards

The extension ensures compliance with web standards by using the manifest.json file to explicitly define required permissions, such as activeTab and storage. These permissions follow the principle of least privilege, requesting only what is essential for the extension's functionality. Additionally, the extension adheres to Chrome's guidelines for creating secure and privacy-friendly extensions, ensuring compliance with the **Chrome Web Store Developer Program Policies**. This approach not only supports seamless distribution through the Chrome Web Store but also reinforces user trust in the extension's reliability and security.



## 9.0 QUALITY INDICATORS

This section evaluates the quality of the project's implementations. It examines whether the outputs meet the agreed objectives and are consistent, reliable, and usable. Additionally, it discusses the potential for generalizing the outcomes to inform future developments and decision-making.

### 9.1 OUTPUTS OF THE PRODUCT

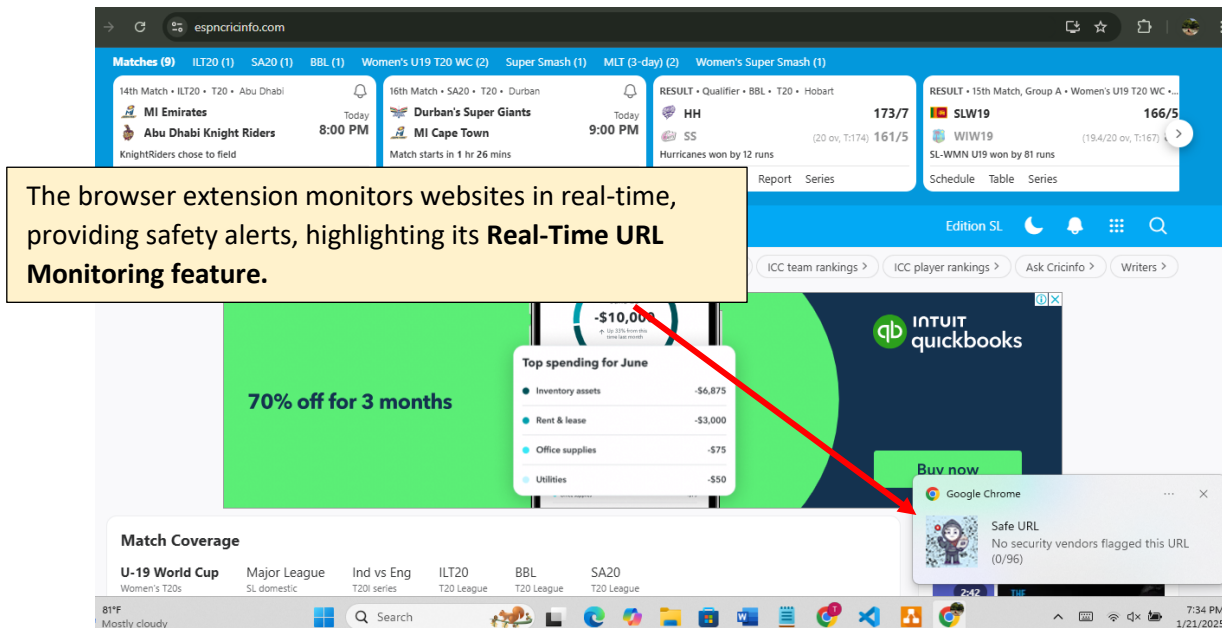


Figure 10: Evidence 1

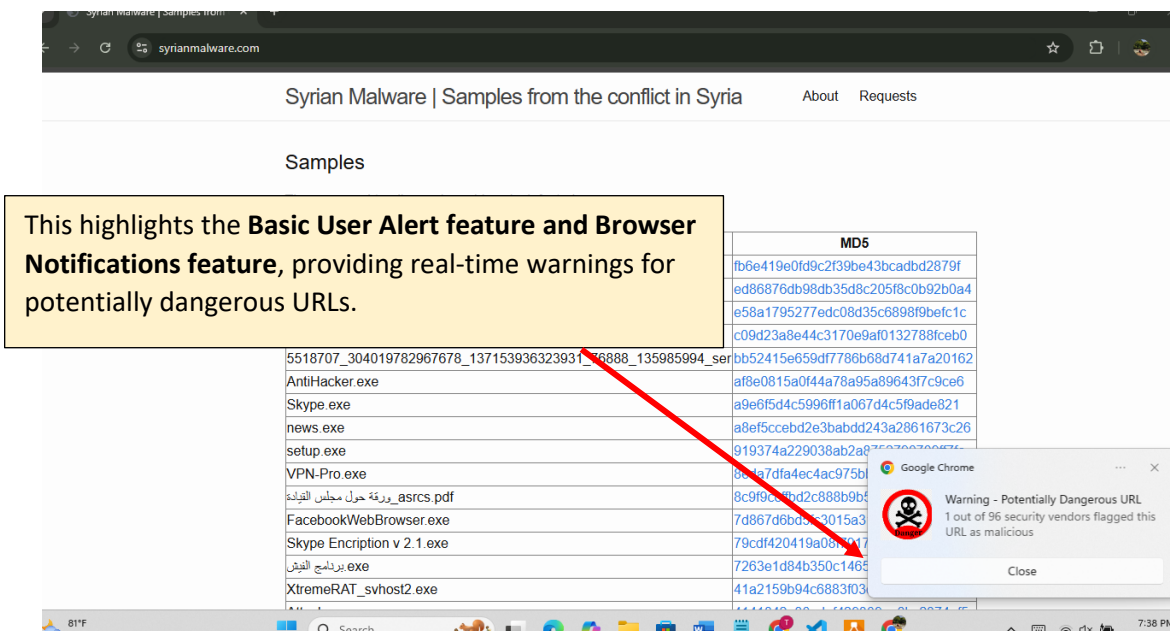


Figure 11: Evidence 2

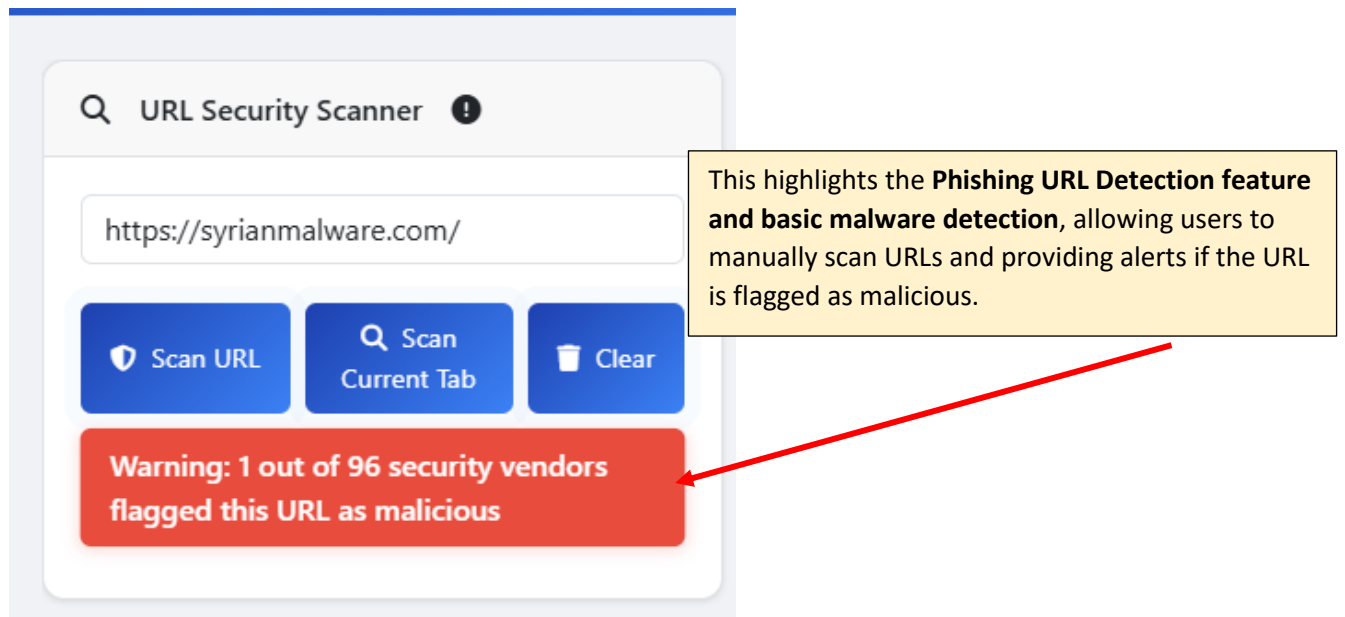


Figure 12: Evidence 3

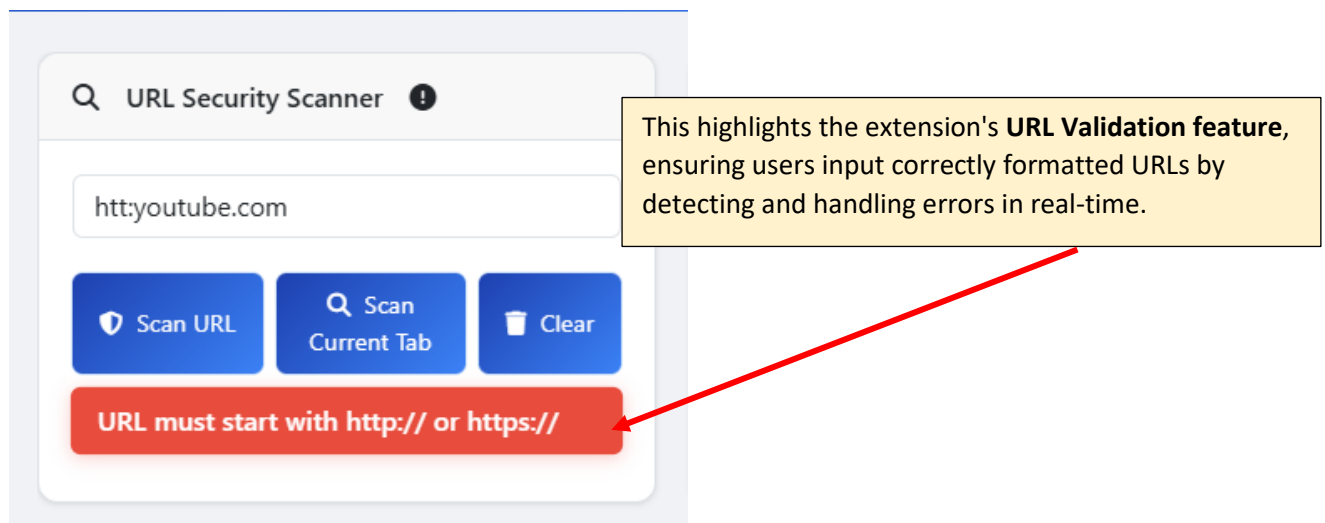


Figure 13: Evidence 4

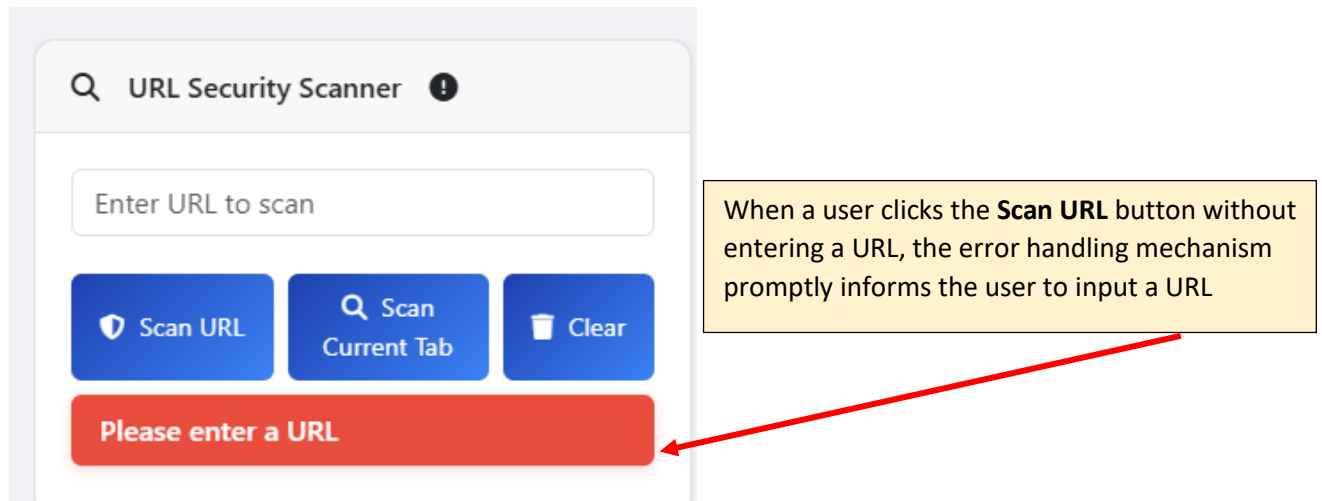


Figure 14: Evidence 5

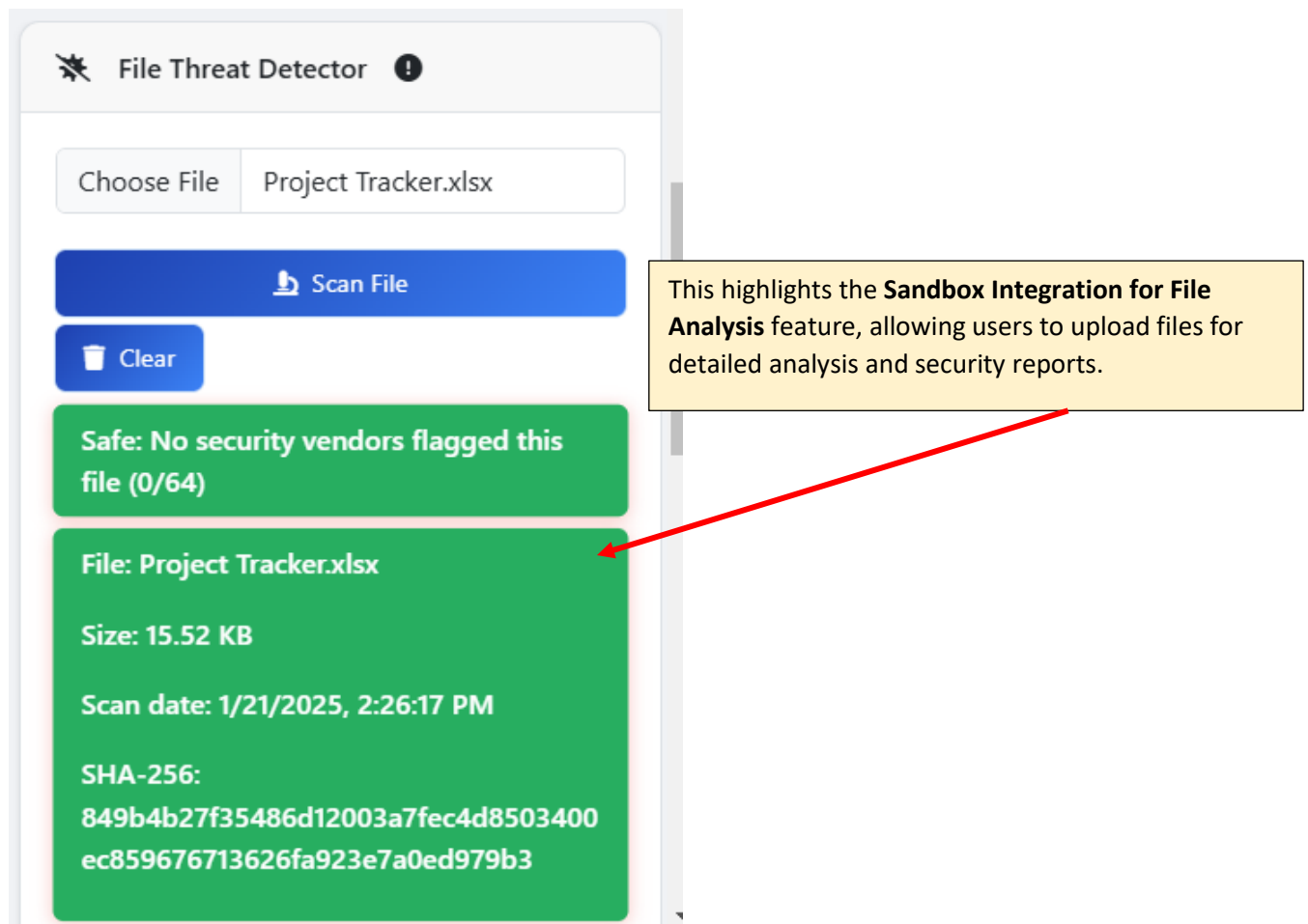
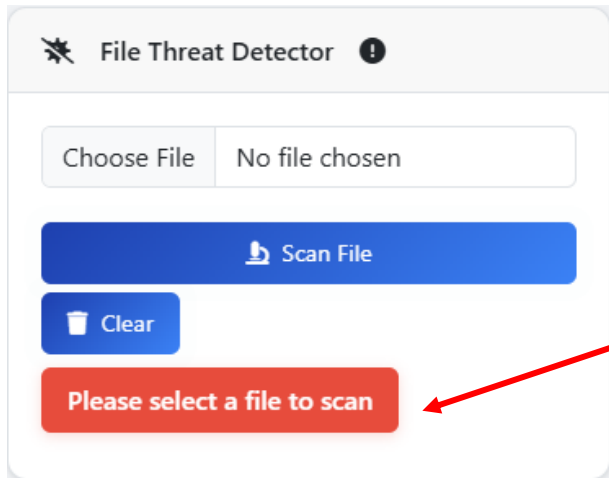
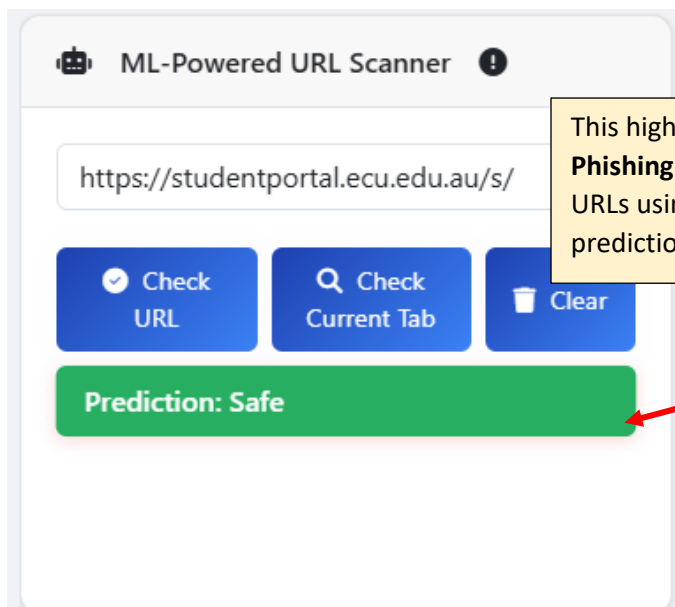


Figure 15: Evidence 6



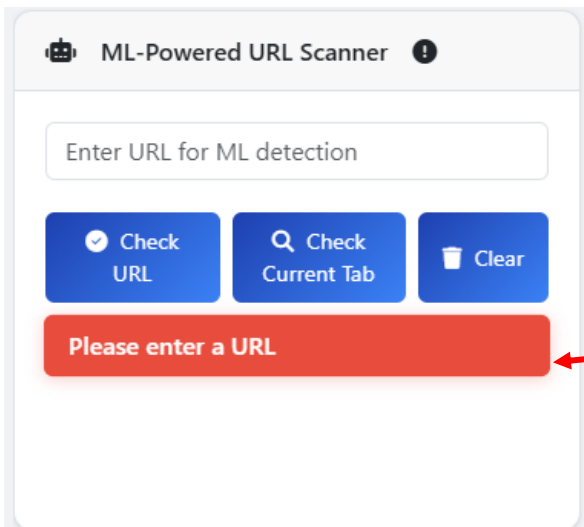
This highlights the **Error Handling**, ensuring that users are prompted with an error message if they attempt to scan a file without selecting one

Figure 16: Evidence 7



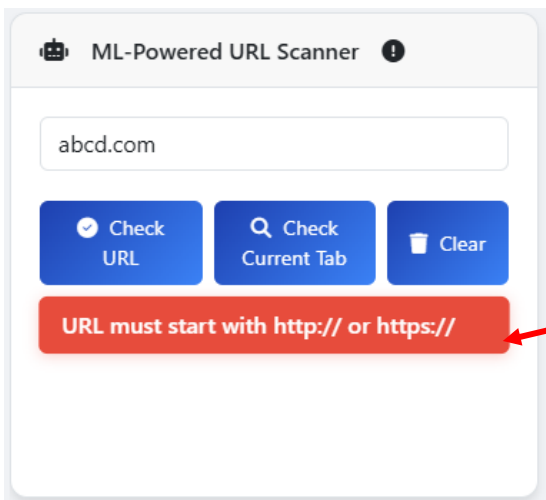
This highlights the **Advanced Machine Learning for Phishing Detection** feature, allowing users to scan URLs using pre-trained ML models for accurate safety predictions.

Figure 17: Evidence 8



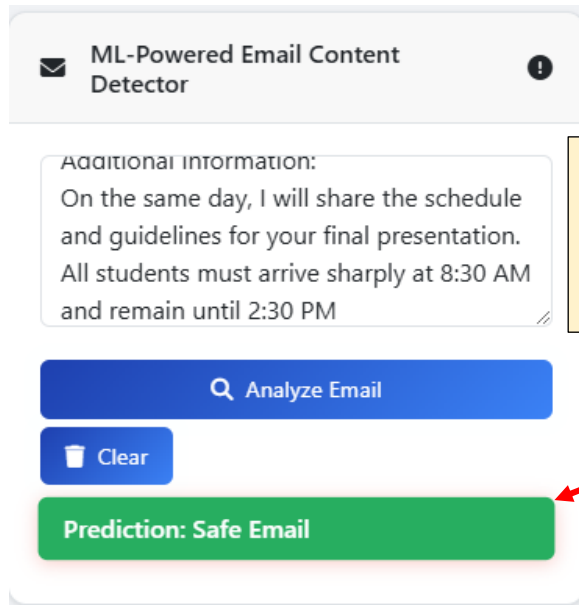
When a user clicks the **Check URL** button without entering a URL, the error handling mechanism promptly informs the user to input a URL

Figure 18: Evidence 9



This highlights the extension's **URL Validation feature**, ensuring users input correctly formatted URLs by detecting and handling errors in real-time.

Figure 19: Evidence 10



ML-Powered Email Content Detector

Additional information:  
On the same day, I will share the schedule and guidelines for your final presentation. All students must arrive sharply at 8:30 AM and remain until 2:30 PM

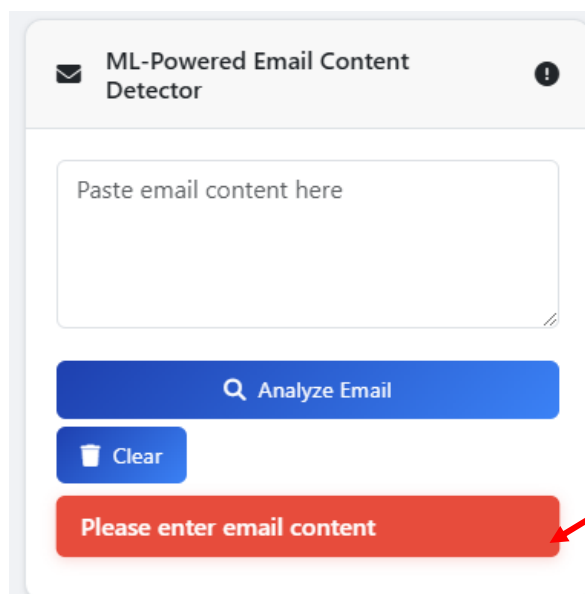
Analyze Email

Clear

Prediction: Safe Email

This highlights the **Email Phishing Detection** feature, allowing users to paste email content or headers for analysis to detect phishing indicators in embedded links or sender details.

Figure 20: Evidence 11



ML-Powered Email Content Detector

Paste email content here

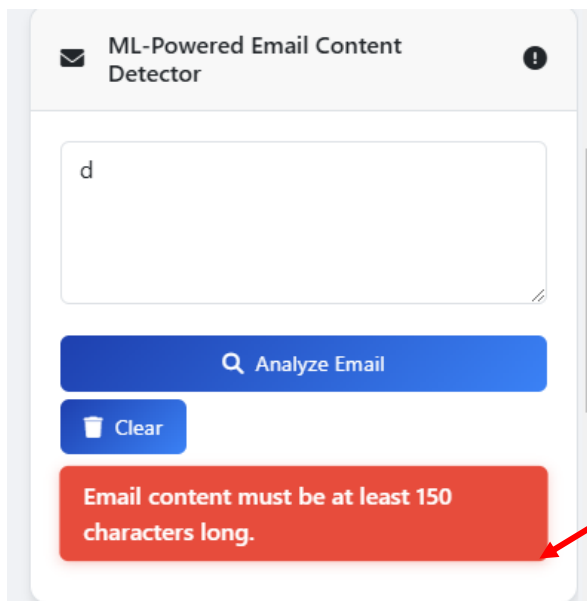
Analyze Email

Clear

Please enter email content

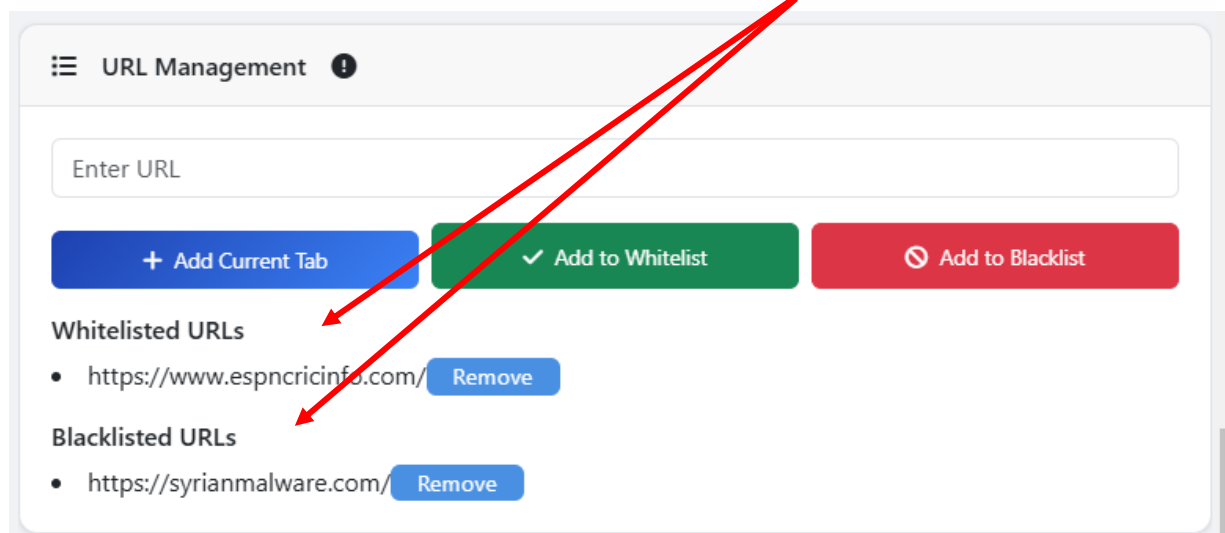
If a user tries to analyze an email without pasting any content, the system promptly displays an error message requesting the user to enter email content.

Figure 21: Evidence 12



If the email content is less than 150 characters, the system displays an error message prompting the user to enter content with at least 150 characters for analysis.

Figure 22: Evidence 13



This highlights the **Whitelist/Blacklist Management** feature, allowing users to add URLs to either a whitelist or blacklist for customized URL handling and security preferences.

Figure 23: Evidence 14

## REPORT: FINAL PROJECT OUTPUTS

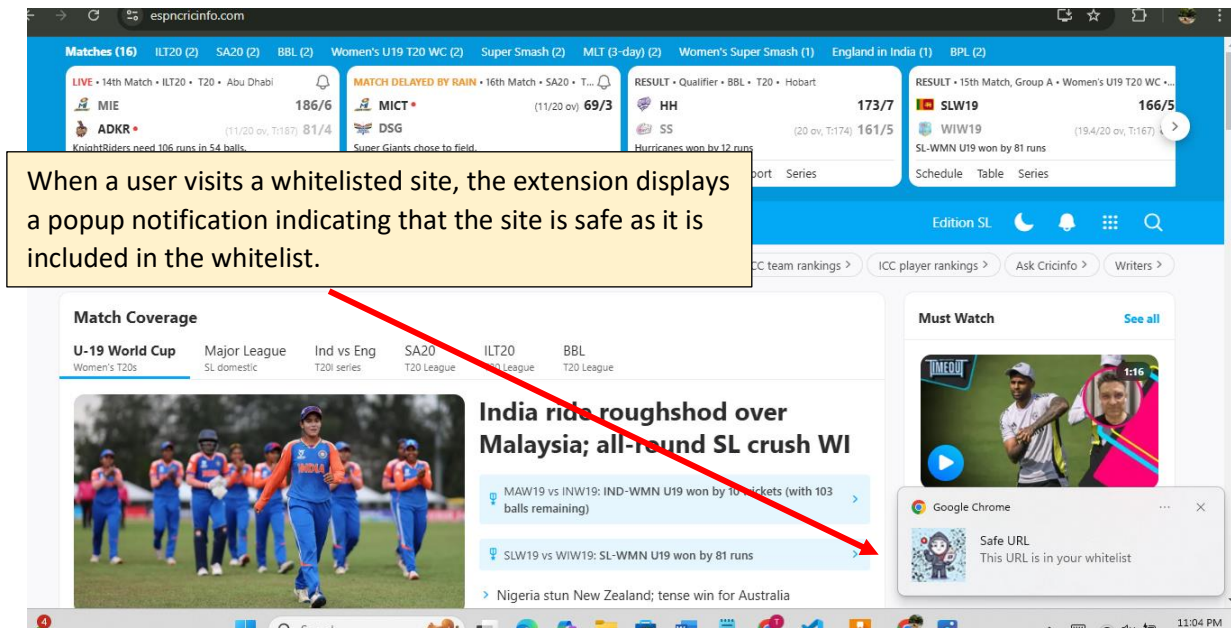


Figure 24: Evidence 15

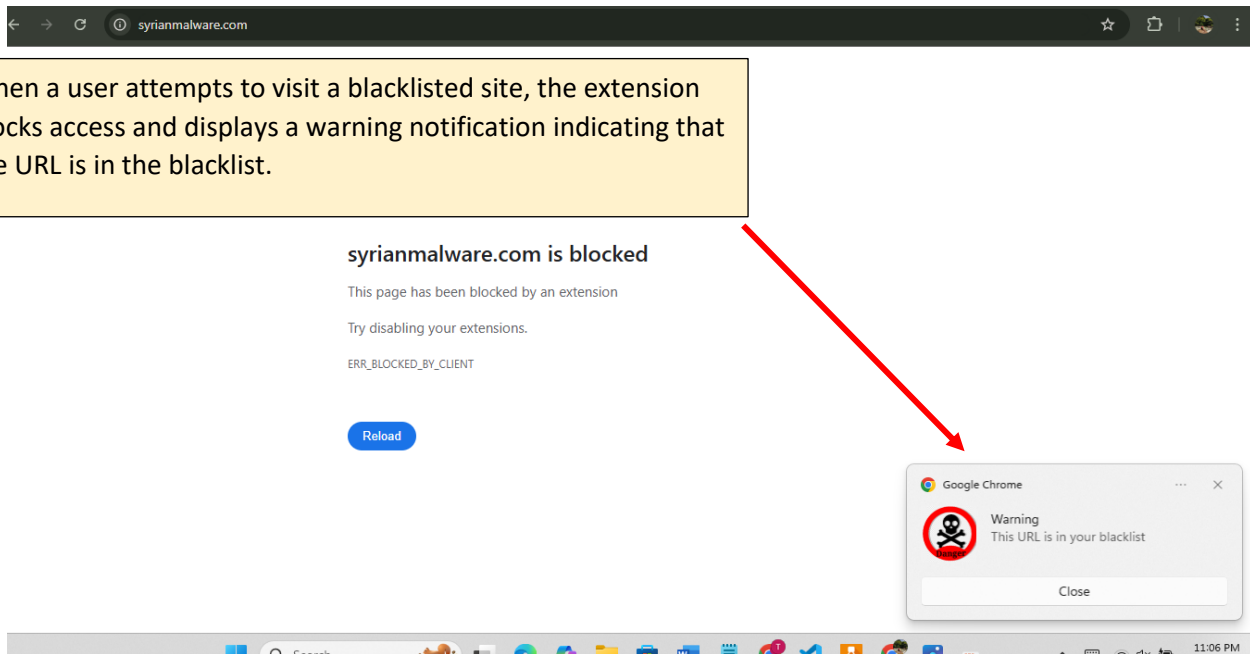


Figure 25: Evidence 16



## REPORT: FINAL PROJECT OUTPUTS

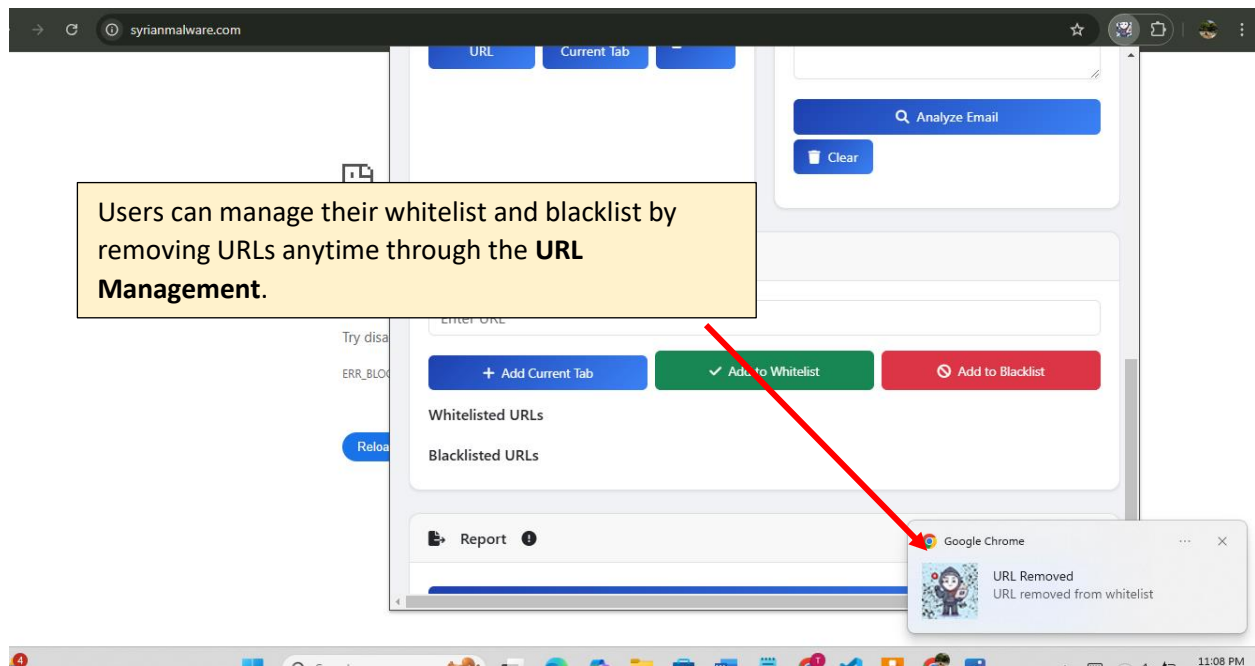


Figure 26: Evidence 17

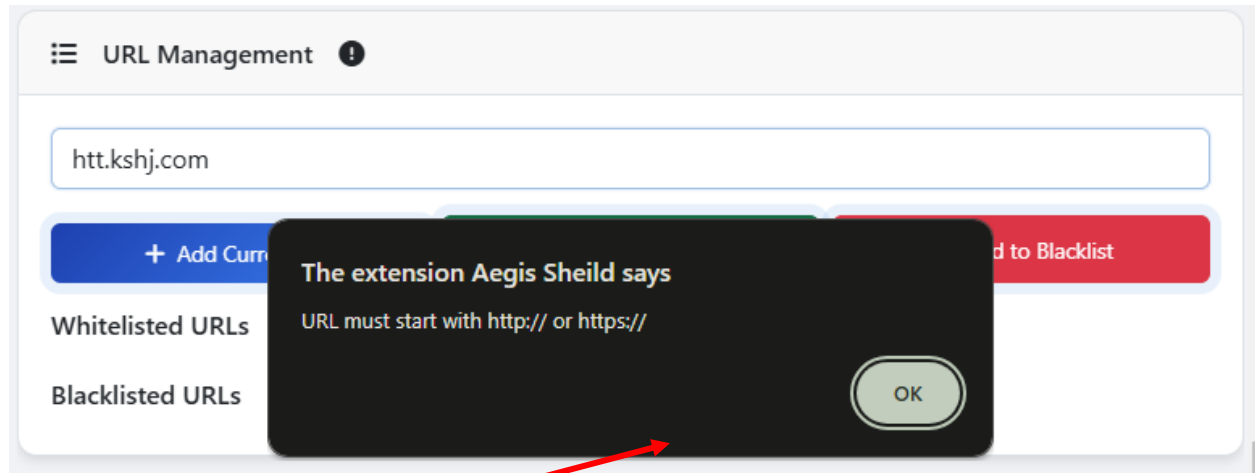


Figure 27: Evidence 18

Users cannot add incorrectly formatted URLs to the **URL Management** feature, as the system validates the input and provides an error message when the URL format is invalid.

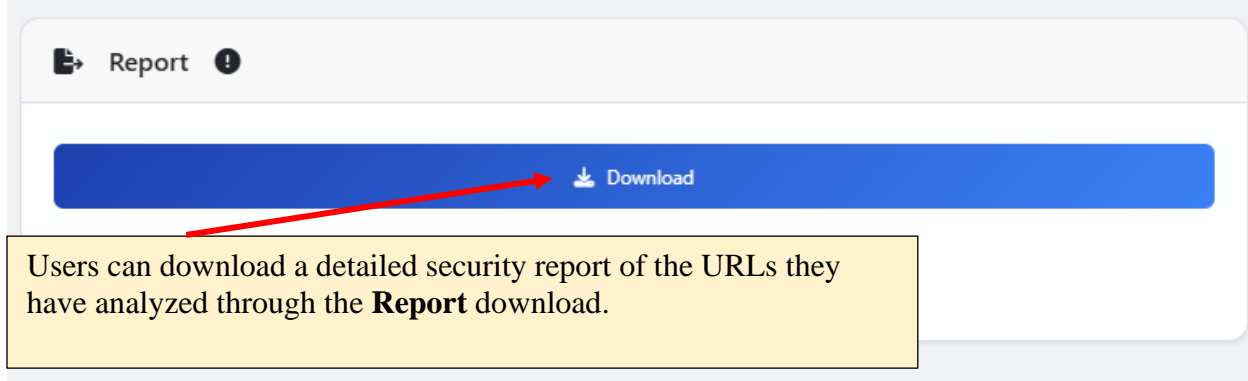


Figure 28: Evidence 19

	A	B	C	D	E	F
1	Timestamp	URL/Content	Category	Analysis Type	Result	Details
2	2025-01-21T14:59:11.04	https://studentportal.ecu.edu.au/s/	URL Scan	Real-Time ML	Safe	Machine learning prediction
3	2025-01-20T13:38:54.28	https://www.youtube.com/	URL Scan	Real-Time ML	Safe	Machine learning prediction
4	2025-01-21T15:05:18.57	Dear Students,	Email	ML Analysis	Safe Email	Email content analyzed
5						

Figure 29: Evidence 20

The downloaded security report provides a detailed summary, including the **timestamp**, **URL or content** analyzed, **category** (e.g., URL Scan or Email), **analysis type** (e.g., Real-Time ML or ML Analysis), **result** (e.g., Safe), and additional **details** about the analysis. This allows users to review and maintain records of their security scans.

The AI-voiced demo video, showcasing the extension's features for clarity and professionalism, is included in the submitted zip file or accessible via the provided link [Demo Video](#).

## 9.2 PERFORMANCE TESTING

The testing of our product was conducted in four rigorous phases to ensure reliability, functionality, and high performance. **Phase 1** focused on foundational features, consisting of **6 test cases** with **36 test scenarios**, providing an initial validation of core functionalities. Subsequent phases (**2, 3, and 4**) expanded the scope, covering **12 test cases** and **61 test scenarios** per phase, addressing both enhancements and new features. Altogether, we tested **40 test cases** across **219 test scenarios**, meticulously validating each component of the product. This comprehensive testing approach not only added significant value to the product but also instilled a high level of confidence in its overall performance and readiness for deployment. The extensive testing ensured that the product meets user expectations and adheres to industry standards, with most features delivering exceptional reliability.

Comprehensive documentation of these testing phases, along with detailed results and methodologies, is provided in the **Testing and Evaluation Report**. For additional details about the entire testing lifecycle, refer to the submitted zip file or follow this link to access [Testing and Evaluation Report](#).

### 9.2.1 FINAL VERSION TESTING (V4)

On **January 20, 2025**, the fourth and final testing phase for the Aegis Shield - Phishing Detection Extension was conducted. This phase involved evaluating all 12 core features developed in the project using a structured approach of test cases and scenarios. The testing aimed to validate the refinements and integrations made since the last iteration, ensuring the extension's robustness and reliability. This round of testing focused on addressing previous limitations and ensuring the extension was ready for deployment.

The table below summarizes the results of Version 4 testing, including the number of scenarios tested, passed scenarios, and success rates for each feature. These metrics showcase the progress made in addressing issues from prior versions and highlight areas where features have achieved optimal functionality.

Requirement name	Test Case ID	No. of Scenarios Tested
Real-Time URL Monitoring	V2_C1	8
Phishing URL Detection	V2_C2	4
Basic URL Alerts	V2_C3	3
Whitelist/Blacklist Management	V2_C4	7
Basic Malware Detection	V2_C5	4
Email Phishing Detection	V2_C6	4
Browser Notifications	V2_C7	5
User-Friendly Interface	V2_C8	6
Severity-Based Alerts	V2_C9	3
Advanced Machine Learning for Phishing Detection	V2_C10	5
Multi-Browser Compatibility	V2_C11	7
Sandbox Integration for File Analysis	V2_C12	5
Total Number of Test Cases Tested: 12		
Total Number of Scenarios Tested: 61		

*Table 7: Overview of Final Version Testing*

## 9.2.2 TEST CASES AND RESULTS FOR FINAL VERSION (V4)

## TEST CASE 1: Real-Time URL Monitoring

Test Case ID	V4_C1	Test case Description			
Version	Version 4	Verify the extension's ability to monitor and alert users about URLs in real-time based on their safety status.			
Testing Functionality		Real-Time URL Monitoring		Tested By	Sudam
Functionality Priority		Must-Have		Revised By	Tanushka
				Test Date	20-Jan-25
				Review Date	20-Jan-25
Number of Scenarios tested		8			
Number of Scenarios Passed		7			
Success Rate %		88%			
S #	Pre- Condition(s)		S #	Test Data	
1	Extension is installed and running.		1	<a href="https://www.youtube.com/">https://www.youtube.com/</a> and <a href="https://syrianmalware.com/">https://syrianmalware.com/</a>	
2	VirusTotal flagged URLs needed		2	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>	
3	-		3	Visit <a href="https://www.youtube.com/">https://www.youtube.com</a> multiple times	
4	-		4	<a href="https://www.espn.com/">https://www.espn.com/</a>	
5	-		5	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>	
6	The URL should be blacklisted.		6	<a href="https://www.cricbuzz.com/">https://www.cricbuzz.com/</a>	
7	-		7	<a href="http://web.simmons.edu/~grovesd/comm244/notes/week2/links">http://web.simmons.edu/~grovesd/comm244/notes/week2/links</a>	
8	-		8	Security Report	
S #	Scenario(s) and Step(s)		Expected Results		Status
1	Visit valid website and monitor for analysis feedback.		URLs are analyzed and marked		Pass
2	Visit URLs flagged as suspicious by VirusTotal.		Alerts for flagged URLs are triggered.		Pass
3	Simulate reaching API call rate limits		Notification informs the user of API issues.		Pass
4	Browse URLs and observe if real-time feedback is prompt.		Results are displayed without delays.		Pass
5	Trigger notifications for malicious URLs		Notifications are provided for malicious URLs.		Pass
6	Visit URLs in the blacklist to test bypassing or blocking.		Blocking Blacklisted sites		Pass
7	Test URLs with different protocols like HTTP and HTTPS.		HTTP is flagged for risks		Fail
8	Monitor if user data or sensitive information is being logged		No user data is stored outside the scope.		Pass

Figure 30: TEST CASE 1: Real-Time URL Monitoring

**TEST CASE 2: Phishing URL Detection**

Test Case ID	V4_C2	Test case Description			
Version	Version 4	Verify the extension's ability to detect and classify URLs as safe, or malicious, and provide user alerts.			
Testing Functionality	Phishing URL Detection	Tested By	Tanushka	Test Date	20-Jan-25
Functionality Priority	Must-Have	Revied By	Sudam	Review Date	20-Jan-25
Number of Scenarios tested	4				
Number of Scenarios Passed	4				
Success Rate %	100%				
S #	Pre- Condition(s)	S #	Test Data		
1	Extension is installed and active.	1	<a href="https://www.youtube.com/">https://www.youtube.com/</a>		
2	VirusTotal flagged URL needed	2	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>		
3	-	3	hts://studentportal.ecu.edu.au/s/		
4	-	4	Visit <a href="https://www.youtube.com">https://www.youtube.com</a> multiple times		
S #	Scenario(s) and Step(s)	Expected Results		Actual Results	Status
1	Submit safe URLs and observe analysis results.	URLs are marked as safe.		As expected	Pass
2	Submit known malicious URLs flagged by VirusTotal.	Malicious URLs are flagged with warnings.		As expected	Pass
3	Submit invalid or incomplete URLs for analysis.	Invalid URLs are rejected with user feedback.		As expected	Pass
4	Simulate VirusTotal API unavailability or timeout.	Fallback mechanism informs users of API issues.		As expected	Pass

Figure 31: TEST CASE 2: Phishing URL Detection

**TEST CASE 3: Basic User Alerts**

Test Case ID	V4_C3	Test case Description			
Version	Version 4	Validate the extension's ability to display real-time alerts for malicious and suspicious URLs			
Testing Functionality	Basic User Alerts	Tested By	Tharuka	Test Date	20-Jan-25
Functionality Priority	Must-Have	Revied By	Sudam	Review Date	20-Jan-25
Number of Scenarios tested	3				
Number of Scenarios Passed	3				
Success Rate %	100%				
S #	Pre- Condition(s)	S #	Test Data		
1	Extension is installed and active	1	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>		
2	URL must be in Blacklist	2	<a href="https://www.espnccricinfo.com/">https://www.espnccricinfo.com/</a>		
3	URL must be in Whitelist	3	<a href="https://www.youtube.com/">https://www.youtube.com/</a>		
S #	Scenario(s) and Step(s)	Expected Results		Actual Results	Status
1	Visit a known malicious URL to trigger an alert.	Malicious URL alert is displayed.		As expected	pass
2	Visit a Blacklisted site	Alert is displayed.		As expected	pass
3	Visit a Whitelisted site	Alert is displayed.		As expected	pass

Figure 32: TEST CASE 3: Basic User Alerts

**TEST CASE 4: Whitelist/Blacklist Management**

Test Case ID	V4_C4	Test case Description											
Version	Version 4	Validate the extension's ability to manage URLs											
Testing Functionality		Whitelist/Blacklist Management				Tested By		Dulaj		Test Date		20-Jan-25	
Functionality Priority		Must-Have				Revised By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		7											
Number of Scenarios Passed		7											
Success Rate %		100%											
S #	Pre- Condition(s)					S #	Test Data						
1	-					1	<a href="https://www.youtube.com/">https://www.youtube.com/</a>						
2	-					2	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>						
3	-					3	-						
4	-					4	-						
5	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a> in Blacklist					5	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>						
6	-					6	<a href="https://www.youtube.com/">hps://www.youtube.com/</a>						
7	-					7	-						
S #	Scenario(s) and Step(s)					Expected Results				Actual Results		Status	
1	Add a trusted website to the whitelist.					Website is successfully added to the whitelist.				As expected		Pass	
2	Add a malicious website to the blacklist.					Website is successfully added to the blacklist.				As expected		Pass	
3	Remove a website from the whitelist.					Website is removed from the whitelist.				As expected		Pass	
4	Remove a website from the blacklist.					Website is removed from the blacklist.				As expected		Pass	
5	Visit a website in the blacklist to verify access is blocked.					Blacklisted website access is blocked.				As expected		Pass	
6	Attempt to add invalid URLs to whitelist/blacklist.					Invalid entries are rejected with feedback.				As expected		Pass	
7	Restart the browser and verify whitelist/blacklist persistence.					List changes are saved and persist after a restart.				As expected		Pass	

Figure 33: TEST CASE 4: Whitelist/Blacklist Management

**TEST CASE 5: Basic Malware Detection**

Test Case ID	V4_C5	Test case Description									
Version	Version 4	Verify the extension's ability to detect malware threats in URLs using the VirusTotal API and provide alerts.									
Testing Functionality		Basic Malware Detection		Tested By		Sudam		Test Date		20-Jan-25	
Functionality Priority		Must-Have		Revied By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		4									
Number of Scenarios Passed		4									
Success Rate %		100%									
S #	Pre- Condition(s)					S #	Test Data				
1	Extension is installed and active.					1	<a href="https://www.youtube.com/">https://www.youtube.com/</a>				
2	-					2	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>				
3	-					3	<a href="https://www.youtube.com/">hps://www.youtube.com/</a>				
4	-					4	Visit <a href="https://www.youtube.com/">https://www.youtube.com/</a> multiple times				
S #	Scenario(s) and Step(s)			Expected Results			Actual Results		Status		
1	Submit a known safe URL for analysis.			Safe URL is marked as safe with no alerts.			As expected		Pass		
2	Submit a known malware-infected URL flagged by VirusTotal.			Malware URL is flagged with an appropriate alert.			As expected		Pass		
3	Submit an invalid or malformed URL for analysis.			Invalid URLs are rejected with an error message.			As expected		Pass		
4	Simulate API timeout and check system behavior.			Graceful fallback message for API failure displayed.			As expected		Pass		

Figure 34: TEST CASE 5: Basic Malware Detection



**TEST CASE 6: Email Phishing Detection**

Test Case ID	V4_C6	Test case Description									
Version	Version 4	Validate the extension's ability to analyze email content or headers, detect phishing indicators									
Testing Functionality		Email Phishing Detection		Tested By		Dulaj		Test Date		20-Jan-25	
Functionality Priority		Must-Have		Reviewed By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		4									
Number of Scenarios Passed		4									
Success Rate %		100%									
S #	Pre- Condition(s)			S #		Test Data					
1	Extension is installed and active.			1		Legitimate email data					
2	phishing email samples are available.			2		phishing email data					
3	-			3		-					
4	-			4		Legitimate email data					
S #	Scenario(s) and Step(s)			Expected Results			Actual Results		Status		
1	Paste content of a legitimate email and analyze.			Safe email content is analyzed and marked as safe.			As expected		Pass		
2	Paste content of a known phishing email for analysis.			Phishing email content is flagged .			As expected		Pass		
3	Submit invalid or empty email content for analysis.			Invalid inputs are rejected with an error message.			As expected		Pass		
4	Analyze a lengthy email with multiple components.			System handles large emails without issues.			As expected		Pass		

Figure 35: TEST CASE 6: Email Phishing Detection

**TEST CASE 7: Browser Notifications**

Test Case ID	V4_C7	Test case Description									
Version	Version 4	Validate the extension's ability to send categorized browser notifications with severity levels									
Testing Functionality		Browser Notifications		Tested By		Sudam		Test Date		20-Jan-25	
Functionality Priority		Must-Have		Revised By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		5									
Number of Scenarios Passed		5									
Success Rate %		100%									
S #	Pre- Condition(s)			S #	Test Data						
1	Extension is installed and active.			1	<a href="https://www.youtube.com/">https://www.youtube.com/</a>						
2	-			2	<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>						
3	-			3	-						
4	-			4	-						
5	-			5	-						
S #	Scenario(s) and Step(s)			Expected Results				Actual Results		Status	
1	Visit a safe URL to trigger a notification.			Notification indicates the URL is safe.				As expected		Pass	
2	Visit a malicious URL to trigger a notification.			Notification clearly warns about the malicious URL.				As expected		Pass	
3	Verify the notification is displayed promptly.			Notifications appear without noticeable delays.				As expected		Pass	
4	Dismiss a notification manually.			Dismissed notifications are removed properly.				As expected		Pass	
5	Verify notifications have consistent design.			Notifications maintain consistent style and icons.				As expected		Pass	

Figure 36: TEST CASE 7: Browser Notifications

**TEST CASE 8: User-Friendly Interface**

Test Case ID	V4_C8	Test case Description			
Version	Version 4	Validate the extension's user interface for clarity, accessibility, and responsiveness			
Testing Functionality	User-Friendly Interface	Tested By	Tanushka	Test Date	20-Jan-25
Functionality Priority	Must-Have	Revised By	Dulaj	Review Date	20-Jan-25
Number of Scenarios tested	6				
Number of Scenarios Passed	5				
Success Rate %	83%				
S #	Pre- Condition(s)	S #	Test Data		
1	-	1	-		
2	-	2	-		
3	-	3	-		
4	-	4	-		
5	-	5	-		
6	-	6	-		
S #	Scenario(s) and Step(s)	Expected Results		Actual Results	Status
1	Hover over a button or feature to check tooltip visibility.	Tooltips are displayed when hovering over features		As expected	Pass
2	Review tooltips for clarity and relevance to the feature.	Tooltips are clear, concise, and relevant.		As expected	Pass
3	Switch to dark mode using the toggle button.	Dark mode is enabled successfully.		As expected	Pass
4	Restart the browser and verify dark mode preference persists.	Dark mode preference persists across sessions.		As expected	Pass
5	Enter data in fields and clear them using the clear button.	Fields are cleared successfully using the button		As expected	Pass
6	Resize the browser window and observe layout adaptability.	Interface adapts responsively to various screen size		Not working	Fail

Figure 37: TEST CASE 8: User-Friendly Interface

**TEST CASE 9: Severity-Based Alerts**

Test Case ID	V4_C9	Test case Description									
Version	Version 4	Validate the extension's ability to provide severity-based alerts and recommended actions for users.									
Testing Functionality		Severity-Based Alerts		Tested By		Tharuka		Test Date		20-Jan-25	
Functionality Priority		Should-Have		Revised By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		3									
Number of Scenarios Passed		2									
Success Rate %		66%									
S #	Pre- Condition(s)			S #		Test Data					
1	Extension is installed and active.			1		<a href="https://www.youtube.com/">https://www.youtube.com/</a>					
2	-			2		<a href="https://syrianmalware.com/">https://syrianmalware.com/</a>					
3	-			3		-					
S #	Scenario(s) and Step(s)			Expected Results			Actual Results		Status		
1	Visit a URL flagged as safe			Alert is displayed			As Expected		Pass		
2	Visit a URL flagged as Malicious			Alert is displayed			As Expected		Pass		
3	Review the alert for suggested actions based on severity.			Alerts include recommended actions for users.			Not working		Fail		

Figure 38: TEST CASE 9: Severity-Based Alerts

## TEST CASE 10: Advanced Machine Learning for Phishing Detection

Test Case ID		V4_C10	Test case Description										
Version		Version 4	Validate the extension's ability to detect phishing patterns using a pre-trained ML model										
Testing Functionality		Advanced ML for Phishing Detection				Tested By		Dulaj		Test Date		20-Jan-25	
Functionality Priority		Could-Have				Revised By		Tanushka		Review Date		20-Jan-25	
Number of Scenarios tested		5											
Number of Scenarios Passed		5											
Success Rate %		100%											
S #	Pre- Condition(s)					S #	Test Data						
1	Extension is installed and active.					1	https://www.youtube.com/ and https://syrianmalware.com/						
2	-					2	<a href="http://freesd1.000webhostapp.com/Star.html">http://freesd1.000webhostapp.com/Star.html</a>						
3	-					3	<a href="http://thelmachan.com.br/images/banners/a607b0c8e7c98759bb45e8a5b1">http://thelmachan.com.br/images/banners/a607b0c8e7c98759bb45e8a5b1</a>						
4	-					4	ht://you'						
5	-					5	Using 5 legitimate and 5 phishing URLs						
S #	Scenario(s) and Step(s)					Expected Results				Actual Results		Status	
1	Submit legitimate URLs for analysis and observe predictions.					Legitimate URLs are correctly identified as safe.				As expected		Pass	
2	Submit known phishing URLs and verify detection results.					Phishing URLs are flagged accurately.				As expected		Pass	
3	Submit URLs with suspicious patterns and evaluate predictions.					Suspicious URLs are flagged appropriately.				As expected		Pass	
4	Submit invalid or incomplete URLs and observe error handling.					Invalid inputs are handled with error messages.				As expected		Pass	
5	Analyze multiple URLs to evaluate model performance and speed.					Model performs efficiently without significant delays.				As expected		Pass	

Figure 39: TEST CASE 10: Advanced Machine Learning for Phishing Detection

**TEST CASE 11: Multi-Browser Compatibility**

Test Case ID	V4_C11		Test case Description										
Version	Version 4		Validate the extension's functionality, feature consistency, and performance across multiple browsers										
Testing Functionality		Multi-Browser Compatibility			Tested By			Tanushka		Test Date		20-Jan-25	
Functionality Priority		Won't-Have			Reviewed By			Dulaj		Review Date		20-Jan-25	
Number of Scenarios tested			7										
Number of Scenarios Passed			5										
Success Rate %			71%										
S #	Pre- Condition(s)				S #	Test Data							
1	Install all the requirement Libraries				1	-							
2	Install all the requirement Libraries				2	-							
3	Install all the requirement Libraries				3	-							
4	Install all the requirement Libraries				4	-							
5	Install all the requirement Libraries				5	-							
6	Install all the requirement Libraries				6	-							
7	Install all the requirement Libraries				7	-							
S #	Scenario(s) and Step(s)				Expected Results				Actual Results		Status		
1	Install and test the extension on Google Chrome				Extension works seamlessly on Chrome.				As expected		Pass		
2	Install and test the extension on Microsoft Edge.				Extension works seamlessly on Edge.				As expected		Pass		
3	Install and test the extension on Brave browser.				Extension works seamlessly on Brave.				As expected		Pass		
4	Install and test the extension on Firefox				Extension works seamlessly on Firefox.				Not working		Fail		
5	Verify that all core features function consistently in Edge.				Features are consistent and functional				As expected		Pass		
6	Verify that all core features function consistently in Brave.				Features are consistent and functional				As expected		Pass		
7	Verify that all core features function consistently in Firefox				Features are consistent and functional				Not working		Fail		

Figure 40: TEST CASE 11: Multi-Browser Compatibility

**TEST CASE 12: Sandbox Integration for File Analysis**

Test Case ID	V4_C12		Test case Description												
Version	Version 4		Validate the extension's ability to upload and scan files for malware using the VirusTotal API												
Testing Functionality		Sandbox Integration for File Analysis			Tested By			Tharuka		Test Date		20-Jan-25			
Functionality Priority		Won't-Have			Revied By			Tanushka		Review Date		20-Jan-25			
Number of Scenarios tested		5													
Number of Scenarios Passed		5													
Success Rate %		100%													
S #	Pre- Condition(s)			S #									Test Data		
1	Extension is installed and active.			1									Valid file		
2	-			2									Malicious file		
3	-			3									File larger than 32 MB		
4	-			4									Checking result with Virustotal		
5	-			5									Valid files		
S #	Scenario(s) and Step(s)			Expected Results					Actual Results		Status				
1	Upload a valid file and observe scan results.			Valid file is scanned successfully with no threats.					As expected		Pass				
2	Upload a file containing malware and verify detection.			Malware file is flagged with appropriate details.					As expected		Pass				
3	Attempt to upload a file larger than the size limit.			Large files are rejected with a size limit warning.					As expected		Pass				
4	Verify the accuracy of scan results for uploaded files.			Scan results are accurate and detailed.					As expected		Pass				
5	Upload multiple files sequentially and observe performance.			System handles multiple file uploads efficiently.					As expected		Pass				

Figure 41: TEST CASE 12: Sandbox Integration for File Analysis

The testing results for **Version 4** highlight significant progress and outstanding performance across the product's features. Out of the 12 developed functionalities, 10 achieved a **100% success rate**, including critical features like **Phishing URL Detection**, **Basic URL Alerts**, **Whitelist/Blacklist Management**, and **Sandbox Integration for File Analysis**, demonstrating their robustness and reliability. Features such as **Real-Time URL Monitoring** and **User-Friendly Interface** showed consistent performance with minor areas for improvement, achieving **88%** and **83% success rates**, respectively. While **Severity-Based Alerts** and **Multi-Browser Compatibility** require further refinement, their current success rates of **66%** and **71%** reflect steady progress since earlier versions. Overall, the comprehensive testing of **40 test cases** across **219 scenarios** in all phases ensures high confidence in the product's functionality and readiness for deployment, with only minor enhancements needed for future updates.

Requirement ID	Requirement name	Success Rate of the Version tested			
		V1	V2	V3	V4
1	Real-Time URL Monitoring	0%	75%	88%	88%
2	Phishing URL Detection	25%	75%	75%	100%
3	Basic User Alerts	Not-Developed	100%	100%	100%
4	Whitelist/Blacklist Management	71%	85%	85%	100%
5	Basic Malware Detection	Not-Developed	75%	100%	100%
6	Email Phishing Detection	100%	100%	100%	100%
7	Browser Notifications	Not-Developed	100%	100%	100%
8	User-Friendly Interface	0%	0%	83%	83%
9	Severity-Based Alerts	Not-Developed	66%	66%	66%
10	Email Content Parsing	Not-Developed	Not-Developed	Not-Developed	Not-Developed
11	Advanced Machine Learning for Phishing Detection	Not-Developed	80%	80%	100%
12	Heuristic URL Analysis	Not-Developed	Not-Developed	Not-Developed	Not-Developed
13	Customizable User Settings	Not-Developed	Not-Developed	Not-Developed	Not-Developed
14	Multi-Browser Compatibility	42%	42%	71%	71%
15	Sandbox Integration for File Analysis	Not-Developed	80%	100%	100%

Table 8: Evaluation of Versions



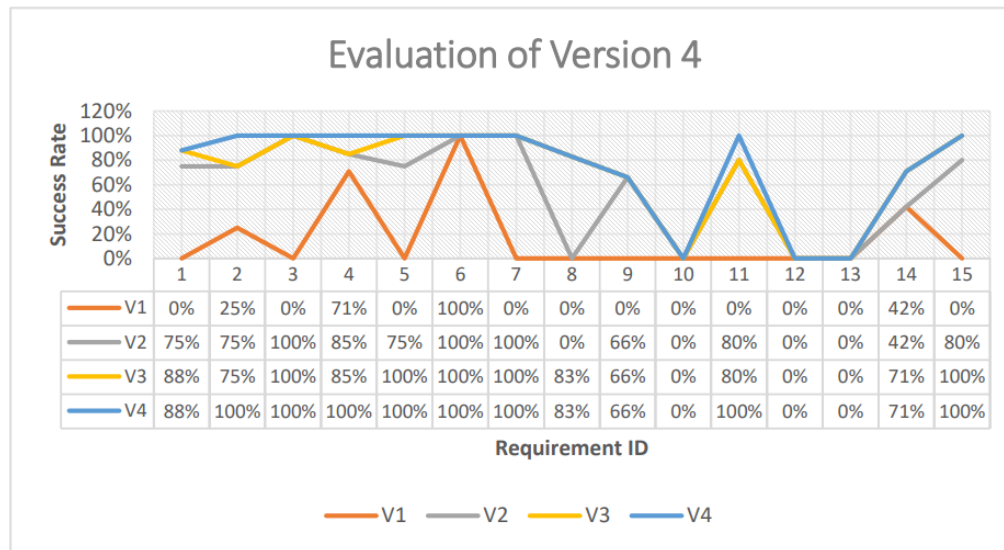


Figure 42: Evaluation of Versions

Comprehensive documentation of these testing phases, along with detailed results and methodologies, is provided in the **Testing and Evaluation Report**. For additional details about the entire testing lifecycle, refer to the submitted zip file or follow this link to access [Testing and Evaluation Report](#).

This detailed analysis adds confidence to the product's readiness and demonstrates the iterative improvements achieved through rigorous testing.

### 9.2.3 IDENTIFIED ISSUES DURING FINAL VERSION TESTING (V4)

While the testing of the final version (Version 4) demonstrated significant improvements and high success rates across most features, a few issues were identified during the process.

These issues are categorized based on their impact and the affected functionalities:

#### 1. Real-Time URL Monitoring

- **Issue:** One test scenario failed where the system did not flag a non-secure (HTTP) website as a potential risk. This occurred due to the lack of checks specifically targeting HTTP URLs, which are inherently less secure than HTTPS websites.
- **Impact:** This failure affected the reliability of the real-time monitoring system, resulting in a success rate of 88% for this feature. Users might not receive proper warnings when visiting potentially insecure websites.
- **Resolution Plan:**
  - Enhance the URL risk analysis logic to include a mandatory check for HTTP websites.

- Implement a specific warning mechanism to alert users about the potential risks associated with non-secure connections.

## 2. Severity-Based Alerts

- **Issue:** One test scenario failed where the alert was generated correctly but the **recommended actions** were not displayed to the user. This impacted the clarity of the alert, as users were left without guidance on how to proceed.
- **Impact:** The feature achieved a success rate of **66%**, reducing the reliability of severity-based alert handling in providing actionable user feedback.
- **Resolution Plan:**
  - Update the alert-handling system to ensure that all alerts include corresponding recommendations and actionable steps for users.
  - Perform additional testing to validate the consistency of severity classifications and their accompanying recommendations.

## 3. Multi-Browser Compatibility

- **Issue:** The extension failed to function properly on the **Firefox browser**. It did not perform as expected due to compatibility issues with Firefox's Web Extension API.
- **Impact:** The success rate for this feature was **71%**, indicating partial functionality on supported browsers (e.g., Chrome and Edge), but limited compatibility with others like Firefox.
- **Resolution Plan:**
  - Analyze the discrepancies in the implementation of browser-specific APIs for Firefox.
  - Modify the extension to ensure compatibility with Firefox's Web Extension standards while maintaining performance on other supported browsers.

## 4. User-Friendly Interface

- **Issue:** Minor UI inconsistencies were observed on smaller screen resolutions. Some interface elements were not properly aligned, impacting the usability of the extension on devices with varying screen sizes.
- **Impact:** The success rate for the interface remained at **83%**, with some users experiencing difficulty interacting with certain elements.
- **Resolution Plan:**
  - Optimize the UI for responsiveness across different screen sizes and resolutions.

- Implement adaptive design techniques and test the interface thoroughly on diverse devices to improve user experience.

### **Impact on Overall Performance**

The final testing results demonstrated that the majority of features performed exceptionally well, with **8 out of 12 features achieving 100% success rates**. The identified issues were specific to certain edge cases and usability aspects, which did not compromise the core functionality of the product but highlighted areas for improvement in user experience and compatibility

## **9.2.4 FUTURE DEVELOPMENTS**

While the current version of the project demonstrates high functionality and meets most of the defined requirements, there is significant potential for further development to enhance its usability, performance, and adaptability.

Below are the areas identified for future improvements and additions:

### **1. Full Multi-Browser Compatibility**

- **Current Status:** The extension works seamlessly on **Chrome, Edge, and Brave** but encountered functionality issues on **Firefox** during testing.
- **Future Plan:**
  - Expand testing and development to resolve compatibility challenges with Firefox.
  - Address browser-specific API implementation differences, particularly for Firefox's WebExtension standards.
  - Ensure seamless functionality across all major browsers, thereby expanding the product's user base and accessibility.

### **2. Enhanced Severity-Based Alerts**

- **Current Status:** Functional, but some test scenarios lacked consistency in providing actionable recommendations.
- **Future Plan:**
  - Refine the severity classification logic for more accurate threat categorization.
  - Include detailed recommendations and action plans with alerts to guide users on mitigating potential risks.
  - Integrate advanced machine learning models to dynamically classify threats and provide intelligent, real-time guidance.

### 3. Development of Heuristic URL Analysis

- **Current Status:** Not yet implemented in the current version.
- **Future Plan:**
  - Introduce heuristic-based analysis to detect suspicious patterns in URLs, such as obfuscated domains, IP-based URLs, or unusual TLDs.
  - Combine heuristic analysis with machine learning-based detection for a hybrid approach, enhancing the overall accuracy of threat identification.

### 4. Customizable User Settings

- **Current Status:** User customization is currently limited to whitelist and blacklist management.
- **Future Plan:**
  - Develop a comprehensive settings panel where users can adjust detection sensitivity, toggle features like phishing detection or file scanning, and customize notification preferences.
  - Enable users to tailor the extension's behavior to suit their specific needs and improve usability.

### 5. Email Content Parsing

- **Current Status:** This feature remains undeveloped, as it was prioritized lower during the initial scope.
- **Future Plan:**
  - Implement email parsing capabilities to analyze email content structure and embedded links for phishing indicators.
  - Add the ability to detect obfuscated or deceptive sender information in emails.
  - Integrate with popular email clients to enable seamless analysis for users.

### 6. Real-Time HTTP Risk Alerts

- **Current Status:** HTTP-specific risks are not comprehensively flagged in the current version of the **Real-Time URL Monitoring** feature.
- **Future Plan:**
  - Introduce dedicated alerts for non-secure (HTTP) websites to inform users of potential security risks associated with unencrypted connections.
  - Incorporate a scoring mechanism to evaluate overall site security and display it to users, enhancing their awareness and decision-making.

## 7. Advanced Reporting Features

- **Current Status:** Basic reporting functionality allows users to download detailed scan results.
- **Future Plan:**
  - Enhance reporting capabilities with graphical visualizations of threats, user behavior trends, and detection insights.
  - Include options to schedule automated reports for periodic reviews, catering to both individual and organizational users.

## 8. Machine Learning Model Enhancements

- **Current Status:** The current models perform well but rely on static training datasets.
- **Future Plan:**
  - Implement dynamic learning capabilities, allowing the models to periodically update with new threat data from trusted sources such as VirusTotal.
  - Enable user feedback integration to refine the models by addressing flagged false positives and false negatives.
  - Enhance the models to adapt quickly to emerging phishing and malware threats.

## 9. VirusTotal Premium API Integration

- **Current Status:** The extension currently uses the free tier of the VirusTotal API, which has rate limitations.
- **Future Plan:**
  - Upgrade to the **VirusTotal Premium API** to increase request limits, improve detection accuracy, and access additional premium features.
  - Ensure faster and more comprehensive threat analysis by leveraging advanced capabilities provided by the premium API.

The proposed future developments aim to address existing gaps and extend the product's capabilities. Enhancing browser compatibility, refining threat detection, introducing heuristic analysis, and upgrading to the VirusTotal Premium API will elevate the extension's performance. Additionally, features like customizable settings, email content parsing, and dynamic machine learning will make the extension more user-focused and adaptable to evolving cybersecurity needs. These advancements will ensure the extension remains a cutting-edge tool for phishing and malware prevention.

## 10.0 SECURITY AND PRIVACY MANAGEMENT

The **Aegis Shield - Phishing Detection Extension** was developed with a clear focus on robust **security and privacy management**, ensuring a secure and trustworthy experience for its users.

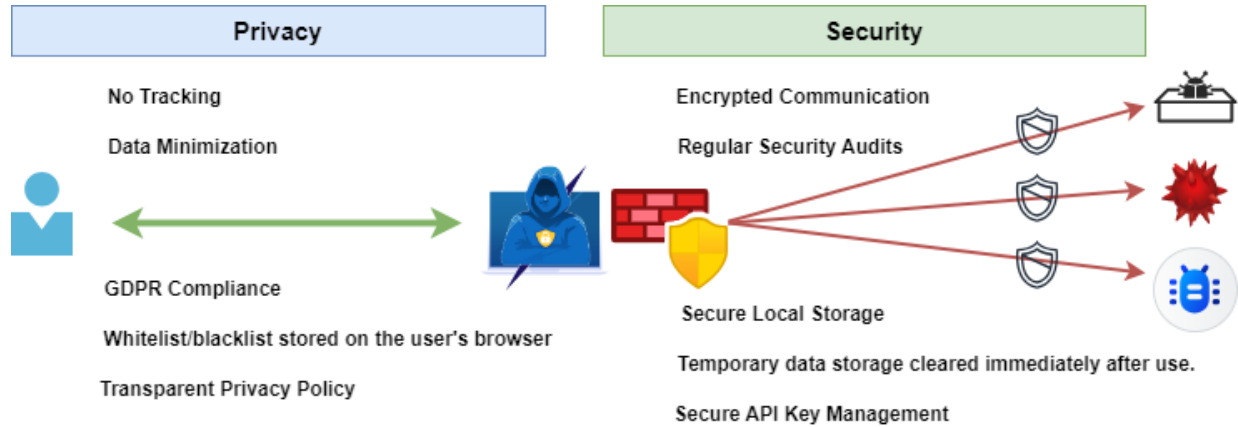


Figure 43: Security and Privacy Management

The core actions taken to manage high-level security while maintaining user privacy are summarized below:

### 1. Data Handling and Processing

#### Actions Taken:

- **Temporary Data Storage:**
  - All user data is processed securely and never stored persistently.
  - Temporary data is cleared immediately after analysis to minimize the risk of unauthorized access.
- **Minimizing Data Collection:**
  - Only essential data (URLs, email content, file hashes) is collected for analysis.
  - No sensitive or unnecessary user data is accessed, stored, or transmitted.
- **Anonymization:**
  - External APIs (e.g., VirusTotal) are only fed anonymized or non-sensitive data (e.g., hashed values of URLs or files) to protect user privacy.

### 2. Encrypted Communication

#### Actions Taken:

- **HTTPS Protocol:**
  - All API communications with services such as VirusTotal are encrypted using HTTPS protocols to ensure data confidentiality.

- **Secure API Key Management:**
  - API keys are securely managed on the backend and are never exposed in client-side code to prevent unauthorized use.
- **End-to-End Encryption:**
  - Data in transit is fully encrypted to protect against interception and tampering.

### 3. Local Storage for User Preferences

#### Actions Taken:

- **Secure Local Storage:**
  - User preferences, such as whitelists and blacklists, are stored locally within the browser using secure mechanisms.
  - This local storage approach ensures that data does not leave the user's device, maintaining control and privacy.
- **No Persistent Storage:**
  - No persistent or cloud-based storage is used for user data, ensuring minimal exposure to breaches.

### 4. Tracking and Monitoring

#### Actions Taken:

- **No User Tracking:**
  - The extension does not monitor, record, or share user browsing behavior or activities beyond its intended functionality.
  - No usage analytics or behavior data is collected, ensuring that user actions remain private and unmonitored.

### 5. Privacy Compliance

#### Actions Taken:

- **GDPR Compliance:**
  - All data handling procedures align with international privacy standards, such as the General Data Protection Regulation (GDPR), ensuring users' data rights are respected.
- **Transparent Privacy Policy:**
  - A clear and concise privacy policy is provided in the Chrome Web Store listing, explaining how data is handled and secured.

### 6. Regular Audits and Testing

#### Actions Taken:

- **Security Audits:**
  - Periodic audits are conducted to identify vulnerabilities and improve security measures.

## 7. Features Supporting Security

### Specific Features:

- **Whitelist and Blacklist Management:**
  - Allows users to add trusted or untrusted websites.
  - URLs in the blacklist are automatically blocked, preventing access to potentially harmful sites.
- **Real-Time Alerts and Notifications:**
  - Immediate warnings ensure users can make informed decisions about malicious URLs or content.
- **Malware Scanning:**
  - Files and URLs are scanned using APIs like VirusTotal without compromising the security of transmitted data.



## 11.0 RISK MANAGEMENT

Our team encountered a situation where we realized the importance of having a proactive approach to handle potential issues that might arise during the project's execution. To address this, we decided to identify and analyze the possible risks that could affect the planned timeline and project deliverables. As an action, we conducted a brainstorming session to evaluate every aspect of the project and compiled a comprehensive list of potential risks. Alongside these risks, we developed corresponding mitigation strategies to ensure that we were well-prepared to manage any challenges effectively. As a result, this proactive approach provided a structured framework to minimize disruptions and ensured a smoother path to project completion.

### Expected Risks

As part of our project proposal, we anticipated potential risks that could arise during the project timeline and developed a proactive risk management approach. To ensure that these risks did not significantly hinder the progress or quality of our project, we proposed a comprehensive **Risk Register** with mitigation strategies to address each identified risk effectively. Through a detailed analysis of project dependencies, resources, and potential challenges, we identified **10 key risks**, covering technical, managerial, and operational aspects. These anticipated risks were supported by well-defined mitigation strategies to ensure that the project remained on track and delivered the expected outcomes.

Risk Rank	Risk Number	Risk Description	Risk Owner	Impact	Risk Mitigation Plan
1	R1	Exceeding API rate limits or third-party API unavailability	Developer	High	Optimize API calls and implement caching; plan for API key rotation.
2	R2	Integration challenges with VirusTotal or other APIs	Developer	High	Perform integration testing early and document API dependencies.
3	R3	Browser compatibility issues with Chrome versions	Developer	Medium	Test extension on multiple Chrome versions; address browser-specific issues.
4	R4	Potential security vulnerabilities in the extension	Security Analyst	High	Conduct security audits and implement secure coding practices.
5	R5	Performance issues slowing down browser operations	Developer	Medium	Optimize extension code and limit resource-intensive operations.
6	R6	Incomplete implementation of Must-Have features	Project Manager	High	Prioritize Must-Have features in early sprints; monitor progress weekly.
7	R7	Insufficient test coverage leading to undetected bugs	QA Specialist	High	Develop comprehensive test plans and use automated testing tools.
8	R8	Skill gaps in team members for tools or APIs	Project Manager	Medium	Allocate time for learning; provide training resources for team members.

9	R9	Timeline delays due to poor time estimates or task dependencies	Project Manager	High	Use Agile boards to monitor tasks; reallocate resources dynamically if delays occur.
10	R10	Low user adoption due to poor usability or unclear value	UI/UX Designer	Medium	Conduct user testing and iterate on UI/UX design based on feedback.

Table 9: Expected Risks

### Experienced Risks

During the course of the project, out of the **10 anticipated risks**, we encountered **3 significant risks** that tested the effectiveness of our risk management strategies.

These risks were addressed promptly, ensuring that they did not compromise the overall quality or delivery timeline of the project. Below is a detailed account of the risks experienced, the challenges they posed, and the actions taken to mitigate them effectively:

#### 1. Exceeding API Rate Limits or Third-Party API Unavailability

- **Risk Encountered:** Our team faced API limitations with the VirusTotal free API, which allows only 4 lookups per minute. This constraint became evident during the real-time URL monitoring and phishing detection processes, where multiple simultaneous requests exceeded the allowed threshold.
- **Actions Taken:**
  - Initially, the team explored the possibility of upgrading to the **VirusTotal premium API**, which offers higher request quotas. However, due to budget constraints and administrative challenges, this option was deemed infeasible.
  - As a resolution, our team decided to continue with the free API but implemented measures to handle the limitations effectively:
    - Added **error messages** to notify users when the API rate limit was exceeded, ensuring transparency in user experience.
    - Adjusted the frequency of API requests within the backend logic to remain compliant with the limitations.
  - **This decision was taken in consultation with the supervisor**, balancing functionality and project constraints while ensuring the extension remained operational.

#### 2. Potential Security Vulnerabilities in the Extension

- **Risk Encountered:** Ensuring the extension was secure and free of vulnerabilities was a critical challenge, as it involved handling user data and interacting with external APIs.

- **Actions Taken:**

- Our team followed **best practices for secure coding**, including sanitizing user inputs, encrypting API communications using HTTPS protocols, and storing sensitive information like API keys in the backend.
- We conducted regular **security audits and testing** to identify potential vulnerabilities. Any identified issues were promptly resolved to safeguard user data and extension functionality.
- The extension's privacy-first design ensured no unnecessary user data was collected or stored, which also reduced potential security risks.
- These actions collectively strengthened the extension's security and compliance with standards such as **ISO/IEC 27001** and **GDPR**.

### 3. Timeline Delays Due to Poor Time Estimates or Task Dependencies

- **Risk Encountered:** Although the majority of tasks progressed smoothly, some tasks exceeded the initially estimated deadlines due to unforeseen challenges in dependency management and task execution.
- **Actions Taken:**
  - Despite these minor delays, **90% of the tasks were completed on time**, reflecting the team's strong commitment and adherence to the planned timeline.
  - For the delayed tasks, the team employed effective resource reallocation and prioritization to ensure that the delays did not affect the overall project timeline or deliverables.
  - Team members actively collaborated and supported each other to address bottlenecks, ensuring that dependencies were managed effectively.
  - As a result, the project was successfully completed and **ready for delivery on January 25, 2025**, as initially planned.

### Overall Outcome

The proactive risk management approach allowed the team to address these challenges effectively. Despite encountering these risks, we adhered to the project timeline and delivered the expected features and functionalities without compromising quality. The measures taken to mitigate these risks not only resolved the immediate issues but also strengthened the overall robustness and reliability of the extension. This experience highlights the importance of risk anticipation, adaptability, and collaboration in project success.

## 12.0 TEAMWORK

This section highlights the collaborative efforts of the team throughout the project. It discusses how tasks were allocated based on individual strengths, the communication methods used to ensure transparency and efficiency, and the professionalism demonstrated in terms of preparedness, commitment, and punctuality. By detailing the team's approach to collaboration and problem-solving, this section underscores the role of teamwork in the successful execution and delivery of the project.

### 12.1 EXCELLENT UNDERSTANDING

Our team, composed of four members, has an exceptional level of understanding and collaboration, which has been instrumental in the successful execution of this project. Unlike teams that are newly formed for specific academic purposes, our bond extends back to the very beginning of our university journey. Since our first year, first semester, we have consistently worked together on various assignments, projects, and extracurricular activities. This long-term collaboration has allowed us to develop a profound understanding of each other's strengths, weaknesses, and working styles.

#### **Strong Team Dynamics**

Over the years, our teamwork has evolved into a highly efficient and supportive dynamic. Each team member brings unique skills and expertise to the table, and our familiarity with one another ensures seamless collaboration. For instance:

- **Tharuka**, known for his strong technical abilities, took the lead on developing the backend systems and integrating APIs, showcasing his expertise in coding and problem-solving.
- **Tanushka**, with a flair for creativity and design, excelled in creating a visually appealing and user-friendly interface for the extension. Her ability to transform complex ideas into intuitive designs significantly enhanced the usability of our product.
- **Sudam**, a meticulous planner, ensured the smooth execution of tasks by focusing on project management and documentation. His organizational skills kept the team on track and ensured deadlines were met.
- **Dulaj**, with his friendly and supportive nature, played a crucial role in fostering a positive team environment. He excelled in quality assurance and testing, using his analytical mindset to identify potential issues and provide valuable feedback to refine the system.

Our strong understanding of each other's skills and working styles played a pivotal role in the success of this project. It eliminated the learning curve typically associated with newly formed teams, allowing us to focus entirely on the project's technical and functional requirements.

Moreover, the mutual trust and camaraderie within the team fostered a supportive environment where everyone could contribute their best work.

## 12.2 COLLABORATIVE APPROACH TO WORK

The success of this project was built on a solid foundation of collaboration and teamwork. From the outset, the team adopted a systematic approach to ensure that all tasks were efficiently managed and completed, leveraging each member's unique skills and abilities. This collaborative approach not only streamlined our workflow but also fostered transparency and accountability throughout the project.

### Task Allocation Based on Abilities

At the start of the project, we collectively listed all the tasks required to achieve our objectives, breaking them down into manageable components. Each task was then assigned to a team member based on their strengths and expertise.

Task No	Task	Required Resources	Estimated Dates	Estimated Deadline	Actual Dates	Actual Deadline	Responsible Person	Notes	Status
1	Define Project Scope	Project management software (MS Excel)	3	25/11/2024	2	25/11/2024	Tanushka	Schedule a meeting with the supervisor for feedback on the project scope.	Completed
2	Set Up Development Environment	VS Code, Chrome Developer Tools, Node.js, APIs	2	30/11/2024	2	30/11/2024	Dulaj	Confirm installation and setup with the supervisor.	Completed
3	Familiarize with Chrome Extension Guidelines	Chrome Developer Guide, API documentation	2	30/11/2024	1	30/11/2024	Tharuka	Supervisor to verify understanding of guidelines.	Completed
4	Schedule the first meeting with supervisor	Ms Teams, Outlook	1	31/11/2024	1	31/11/2024	Sudam	Confirm the supervisor's availability to have a meeting	Completed
5	1st supervisor meeting	Ms Teams	1	2/12/2024	1	2/12/2024	Team	Discuss the project idea with the supervisor.	Completed
6	Create a project Proposal	MS Word	5	8/12/2024	4	8/12/2024	Team	Finalize the scope and complete the project proposal	Completed
7	Schedule the Second meeting with supervisor	Ms Teams, Outlook	1	9/12/2024	1	9/12/2024	Sudam	Confirm the supervisor's availability to have a meeting	Completed
8	2nd supervisor meeting to get the approval for the proposal	Ms Teams, MS Word	1	10/11/2024	1	10/11/2024	Team	Got the supervisor approval	Completed
9	Get the feedback from Miss Ann for the proposal	MS word	1	13/12/2024	1	13/12/2024	Team	Got the feedback	Completed
10	Finalize the proposal based on the feedback	MS Word	2	13/12/2024	1	13/12/2024	Team	Finalized the Proposal	Completed
11	Create and sign the Group Contract	MS Word	2	14/12/2024	1	14/12/2024	Team	Created the team agreement	Completed
12	Project Proposal and Group Contract Submission	Canvas	1	14/12/2024	1	14/12/2024	Sudam	Submitted both Proposal and group contract	Completed
13	Setting Up a Repository on GitHub to share the Codes	GitHub	1	14/12/2024	1	14/12/2024	Tanushka	All members are given access to the Repo	Completed
14	Design the Basic UI , Prototype	Adobe Illustrator, HTML, CSS	3	14/12/2024	3	14/12/2024	Tharuka	Discuss the overall structure and get the ideas	Completed
15	Conduct a research UI design best practices	Figma , Illustrartor, Canva	2	16/12/2024	2	16/12/2024	Dulaj	Schedule a discussion on findings	Completed
16	3rd Supervisor Meeting	MS Teams	1	16/12/2024	1	16/12/2024	Team	Discuss the progress	Completed
17	Design wireframes and prototypes	Figma , Illustrartor, Canva	2	18/12/2024	2	18/12/2024	Sudam	Share designs with the supervisor for feedback	Completed
18	Develop core UI elements	VS code and Anaconda	1	20/12/2024	1	20/12/2024	Tharuka	Schedule a discussion on backend integration challenges	Completed
19	Conduct a research on API	Vrusrtotal API	1	21/12/2024	1	21/12/2024	Tanushka	Share API research outcomes	Completed
20	4th Supervisor Meeting	Ms Teams	1	23/12/2024	1	23/12/2024	Team	Review progress	Completed
21	Dataset preprocessing and cleaning	Jupyter ,VS code	3	27/12/2024	3	27/12/2024	Tanushka	Share preprocessing steps	Completed
22	Split dataset into training and testing sets	Jupyter ,VS code	2	29/12/2024	2	29/12/2024	Sudam	Schedule a session to discuss data splits and any concerns	Completed

Figure 44: Task Allocation Among Team Members

The task allocation details (project tracker), is available in the submitted zip file or can be accessed via the provided link [Project Tracker](#).

### Maintaining a Shared Workspace

To facilitate collaboration, we established a **shared folder** accessible to all team members, our lecturer, and our project supervisor. This folder served as a centralized repository where we uploaded completed tasks, drafts, and resources. Every time a task was completed, it was immediately updated in the folder, allowing everyone involved to track progress in real time. This system not only promoted transparency but also made it easier for our supervisor and lecturer to monitor our work and provide timely feedback.

Tharuka GUNASEKARA > SRI-CSG3101.2 - phishing detection extension

Name	Modified	Modified By	File size	Sharing
Approved project proposal	December 11, 2024	Sudam PULLAPERU	1 items	Shared
Group Contract	December 14, 2024	Sudam PULLAPERU	1 items	Shared
Meeting Details	January 6, 2025	Sudam PULLAPERU	1 items	Shared
Meeting Recordings	January 6, 2025	Sudam PULLAPERU	1 items	Shared
Project Tracker	December 9, 2024	Sudam PULLAPERU	1 items	Shared
Timeline	December 9, 2024	Sudam PULLAPERU	1 items	Shared
applied-01.jpg	December 9, 2024	Tharuka GUNASEK	1.05 MB	Shared
Project draft.docx	December 9, 2024	Sudam PULLAPERU	17.9 KB	Shared

Figure 45: Shared folder environment

### MS Teams for Communication and Updates

In addition to the shared folder, we maintained active communication through our **Microsoft Teams group**. This platform was used to share updates, discuss progress, and address any challenges that arose. By regularly updating the group with task statuses, we ensured that everyone remained aligned with the project's goals and timelines. The use of MS Teams also made it easy to coordinate with our supervisor and seek guidance whenever necessary.

SRI-CSG3101.2 - phishing detection extension Chat Shared

Sudam PULLAPERUMA Group Progression Summary What has been completed so far Project draft created Draft proposal created 1st supervisor meeting done Project tracker created to track the important project

1/14 9:35 PM

7th supervisor meeting done

The recording has uploaded to the shared folder

Meeting Number	Date	Link for the recording
1	Monday, December 2, 2024	<a href="https://edithcowanuni-my.sharepoint.com/_v/p/personal/sp">https://edithcowanuni-my.sharepoint.com/_v/p/personal/sp</a>
2	Tuesday, December 10, 2024	<a href="https://edithcowanuni-my.sharepoint.com/personal/telving">https://edithcowanuni-my.sharepoint.com/personal/telving</a>
3	Monday, December 16, 2024	<a href="https://edithcowanuni-my.sharepoint.com/personal/gtgunar">https://edithcowanuni-my.sharepoint.com/personal/gtgunar</a>
4	Monday, December 30, 2024	<a href="https://edithcowanuni-my.sharepoint.com/personal/gtgunar">https://edithcowanuni-my.sharepoint.com/personal/gtgunar</a>
5	Sunday, January 5, 2025	<a href="https://edithcowanuni-my.sharepoint.com/personal/dwajee">https://edithcowanuni-my.sharepoint.com/personal/dwajee</a>
6	Monday, January 6, 2025	<a href="https://edithcowanuni-my.sharepoint.com/personal/telving">https://edithcowanuni-my.sharepoint.com/personal/telving</a>
7	Tuesday, January 14, 2025	<a href="https://edithcowanuni-my.sharepoint.com/personal/telving">https://edithcowanuni-my.sharepoint.com/personal/telving</a>

Tishantha ELVITIGALA 1/14 9:37 PM

TE

Progress so far

Figure 46: MS Teams used to share updates

## Zoom Meetings to Prepare the Presentation

We conducted regular Zoom meetings to prepare for the team presentations. These meetings provided a platform for collaborative brainstorming, where we could refine our slides and ensure that all key points were effectively covered. During the sessions, each team member shared their slides via Zoom and practiced delivering their sections. This allowed us to provide constructive feedback to one another and ensure consistency and flow across the presentation. These meetings not only helped us polish our presentation skills but also fostered a sense of unity and teamwork as we worked towards a common goal. The use of Zoom proved instrumental in coordinating our efforts, especially in situations where in-person meetings were not feasible.

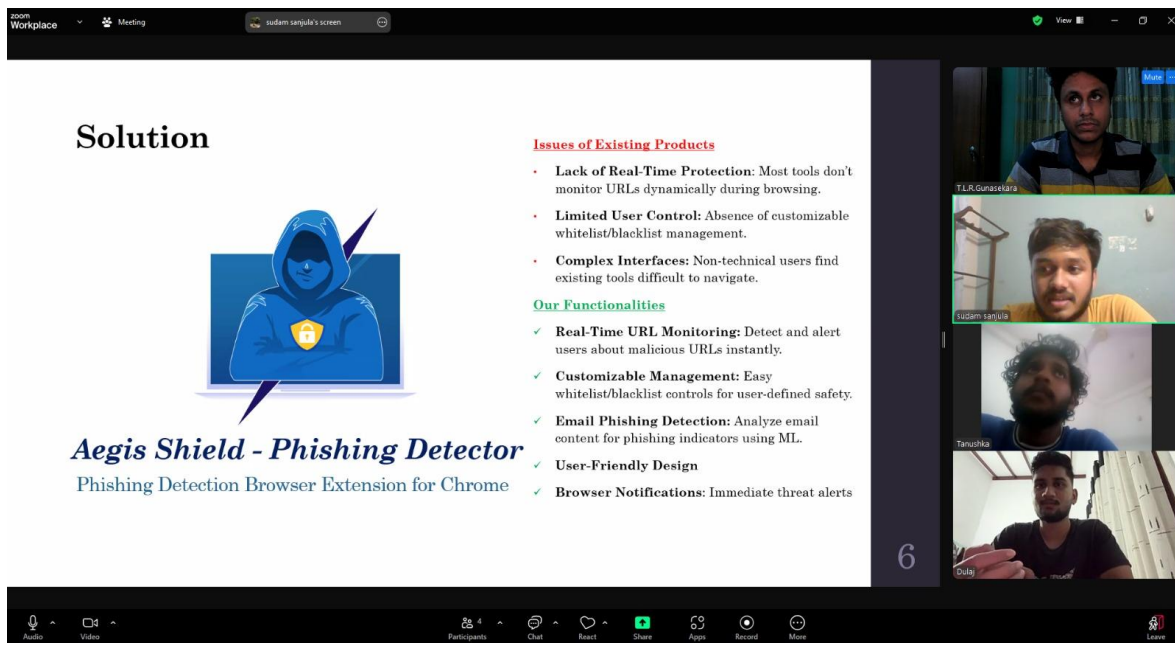


Figure 47: Practicing the presentation using Zoom meetings

## Real-Time Progress Monitoring

The combination of a shared folder and frequent updates in MS Teams allowed the entire team to stay informed about the project's status at all times. Whether it was uploading a completed module, sharing testing results, or updating documentation, every team member had access to the latest developments. This real-time visibility into the project's progress helped us maintain momentum and quickly address any bottlenecks.

## Fostering Team Accountability

By clearly defining tasks and maintaining open communication, we created an environment where every member felt accountable for their responsibilities. The shared folder and MS Teams updates not only kept us on track but also encouraged mutual support. If a team member faced a challenge, others were readily available to offer assistance, ensuring that no task was delayed.

## 12.3 COMMUNICATION FREQUENCY AND CLARITY

Effective communication was a cornerstone of our project's success. Throughout the project lifecycle, the team prioritized consistent and clear communication to ensure that all members, our lecturer, and our project supervisor remained aligned on progress, challenges, and deliverables. By leveraging multiple communication tools and platforms, we established an efficient system to share updates, clarify doubts, and foster collaboration. This section outlines the various communication channels used and their impact on the project's execution.

### Microsoft Teams for Updates

Microsoft Teams was primarily used to communicate with our lecturer and provide regular updates on project progress. The team utilized MS Teams to share significant milestones, discuss feedback from the lecturer, and address any challenges raised during these interactions. With a dedicated group created for the project, every member was kept informed about updates in real-time. This platform also enabled quick decisions and easy access to team discussions, ensuring no delays in addressing issues. Supervisors and lecturers were also included in this group, enabling seamless feedback and guidance.

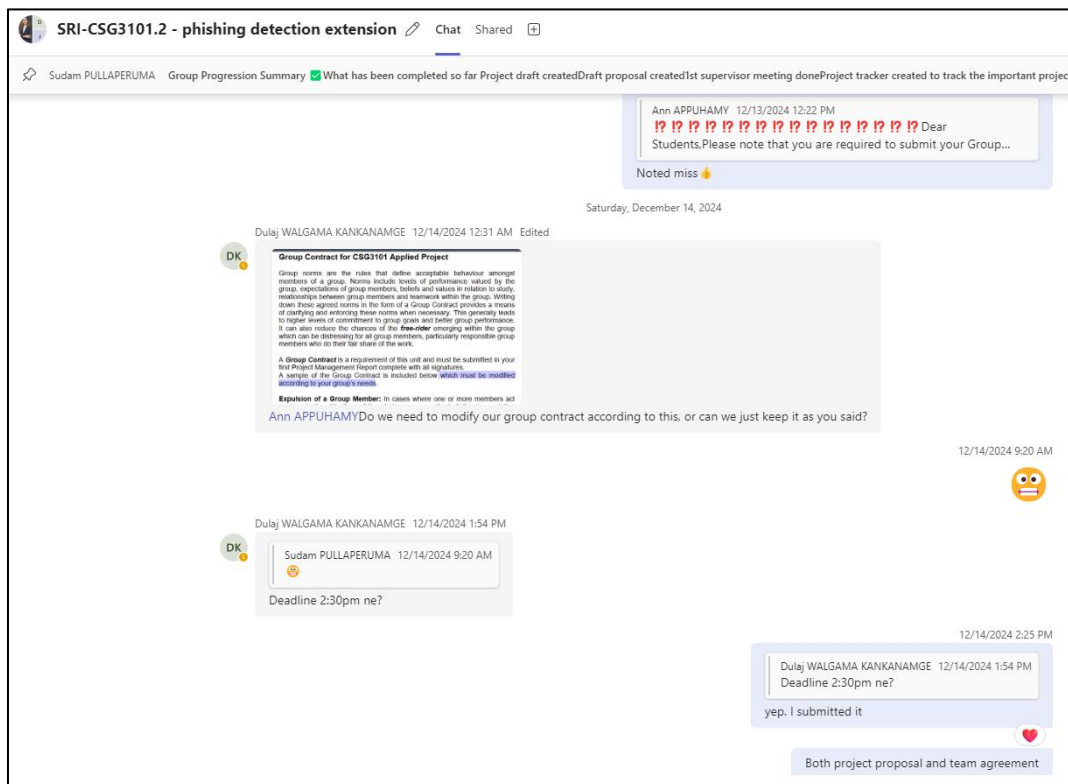


Figure 48: Communication VIA MS Teams



## Zoom for Regular Meetings

Zoom was used to conduct regular team meetings to discuss the project's progress, address challenges, and collaboratively review key deliverables. These meetings included in-depth discussions about tasks, deliverables, and timelines. Additionally, zoom meetings were crucial for reviewing and refining documentation. When a team member completed a document, it was shared during the meeting, allowing the team to review it collectively for errors or suggestions, ensuring high-quality outputs. Zoom was also instrumental in practicing and refining our presentations, with team members sharing their screens to review slides, provide feedback, and rehearse delivery. These collaborative sessions not only improved the overall quality of our work but also strengthened the team's cohesion and alignment.

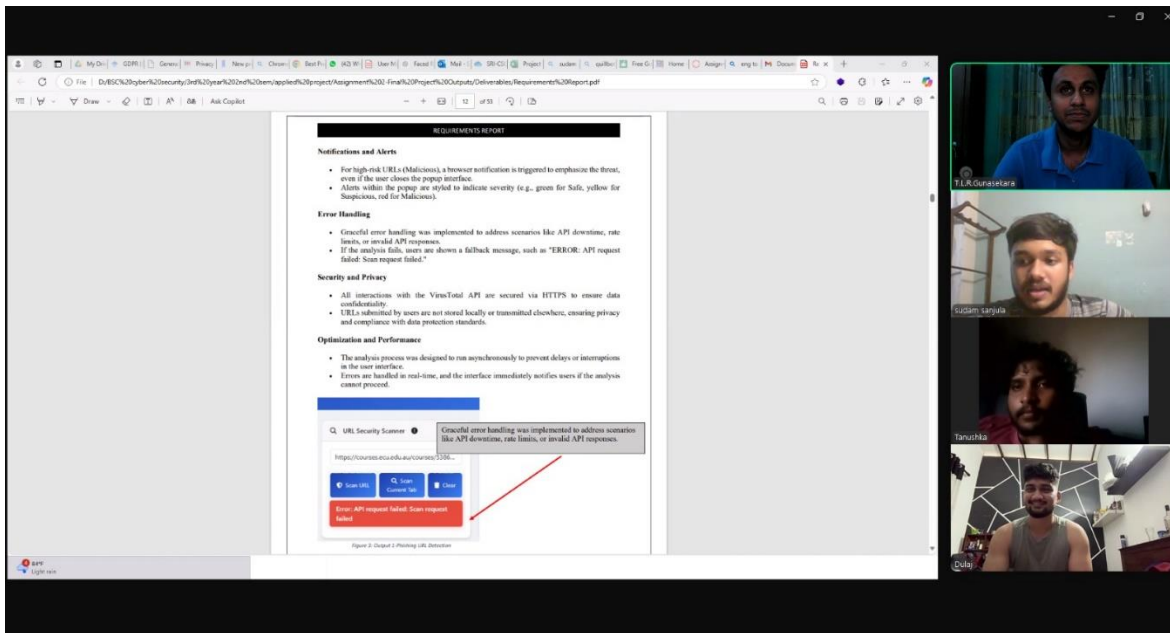


Figure 49: Reviewing one of the documents

## Emails for Formal Communication

Email played a vital role in ensuring formal and structured communication throughout the project. Both university-provided email accounts and personal email addresses were used for sharing important documents, obtaining approvals, and providing access to shared resources. Additionally, emails were used to share Google Docs for collaborative editing of documents, allowing team members to work together seamlessly on project deliverables. The team also utilized email to manage and share access to the GitHub repository, ensuring smooth collaboration on code management. Emails served as a secure and reliable means of communication, even when interacting with external parties, such as the VirusTotal team, to address technical queries or issues.

## REPORT: FINAL PROJECT OUTPUTS

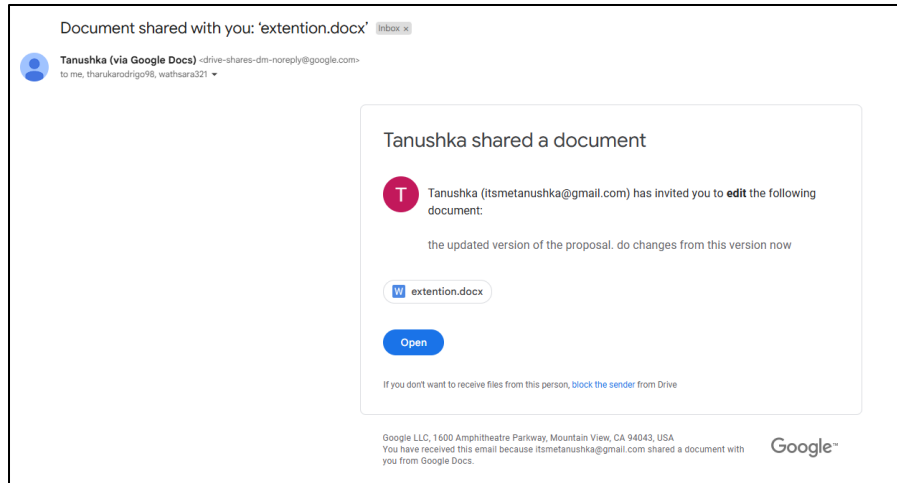


Figure 50: Giving editing access to the team on one of the deliverables through Personal mail

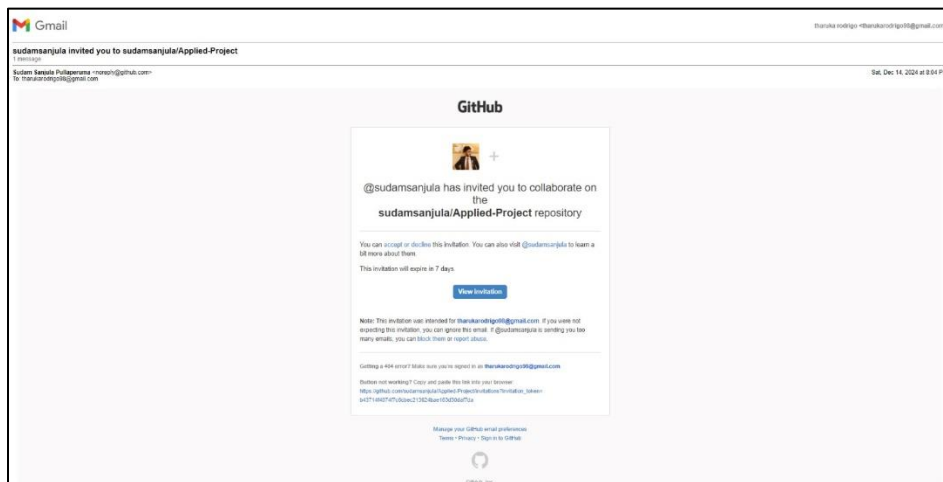


Figure 51: Inviting team members to a GitHub repository

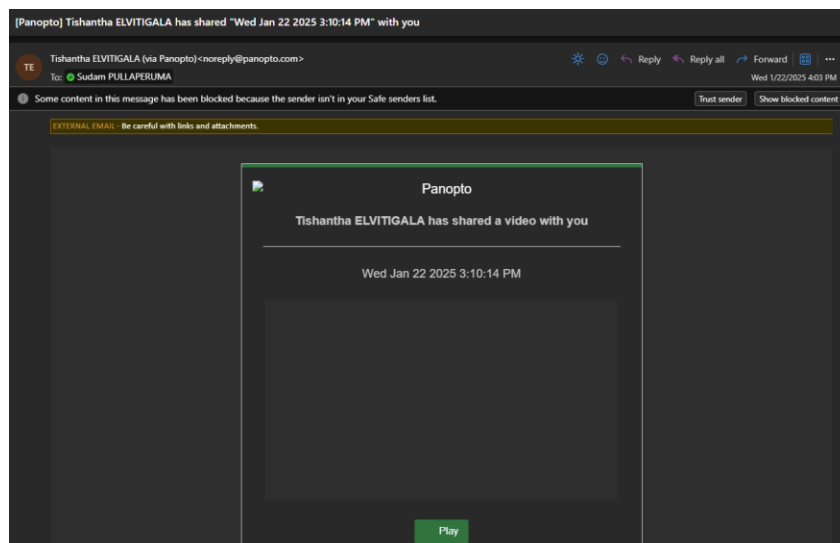


Figure 52: Sharing Panopto recorded demo video to review

### Weekly Meetings with the Supervisor

In addition to internal team discussions, weekly meetings were conducted with our supervisor to provide progress updates and seek guidance. These sessions served as an opportunity to showcase completed work, discuss any challenges faced, and gather constructive feedback. The supervisor's inputs often provided new perspectives and actionable ideas, which were promptly integrated into the project. This iterative feedback loop ensured that the project stayed aligned with its objectives and maintained a high standard of quality.

A		B
Meeting Number	Date	Link for the recording
1	Monday, December 2, 2024	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
2	Tuesday, December 10, 2024	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
3	Monday, December 16, 2024	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
4	Monday, December 30, 2024	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
5	Sunday, January 5, 2025	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
6	Monday, January 6, 2025	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
7	Tuesday, January 14, 2025	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>
8	Wednesday, January 22	<a href="https://edithcowanuni-my.sharepoint.cc">https://edithcowanuni-my.sharepoint.cc</a>

Figure 53: Supervisor Meeting Recordings

By employing a multi-channel communication strategy, the team ensured that all members were consistently informed and aligned with project objectives. The combination of MS Teams for instant updates, zoom for collaborative discussions, email for formal exchanges, weekly supervisor meetings for guidance, and the shared folder for progress tracking allowed us to maintain transparency, clarity, and efficiency throughout the project. These communication practices were instrumental in meeting our deliverables on time and ensuring the quality of our final output.

## 12.4 DEMONSTRATION OF PROFESSIONALISM

Professionalism was a key driving factor behind the success of this project, evident through the team's preparedness, commitment, and punctuality throughout every stage of execution and delivery. By adhering to a structured approach and maintaining high standards of communication and collaboration, the team ensured that all deliverables met expectations while staying aligned with the project timeline.

### Preparedness

- **Thorough Planning:** At the outset of the project, the team engaged in detailed planning, dividing tasks based on individual strengths and ensuring every aspect of the project was accounted for. The use of tools like a Gantt chart and a project tracker ensured clarity and systematic progression. Time was also allocated for learning new tools and frameworks,

such as the VirusTotal API and Chrome Extensions, to ensure smooth implementation during development.

- **Research and Preparation:** The team conducted research on phishing detection techniques and other extensions to better understand the landscape and refine the project scope. This preparation enabled informed decision-making throughout the project lifecycle.
- **Meeting Readiness:** Weekly meetings with the supervisor were well-prepared, with team members presenting progress updates, challenges, and solutions. This proactive approach ensured that feedback was implemented efficiently.

### **Commitment**

- **Dedication to Roles and Responsibilities:** Each team member demonstrated a high level of commitment by taking ownership of their assigned tasks. Team members collaborated to address bottlenecks and ensured the successful delivery of every planned task.
- **Problem-Solving Mindset:** When challenges arose, such as API rate limits and timeline adjustments, the team approached them with resilience and adaptability. For example, the team devised a workaround for the API limitations by integrating error handling and optimizing request frequencies.
- **Extra Effort for Quality:** The team went above and beyond during testing and documentation phases, dedicating additional time to refining outputs. For instance, collaborative Zoom meetings were used to review documents and identify improvements.

### **Punctuality**

- **Timeline Adherence:** Despite minor adjustments to certain intermediate tasks, 90% of the project activities were completed as per the planned timeline. These adjustments did not affect the final delivery, and the project was fully ready for submission by January 25, 2025, as planned.
- **On-Time Deliverables:** Drafts, testing reports, and documentation were consistently submitted on time, enabling iterative improvements. The final presentation materials were prepared and rehearsed in advance to ensure a polished delivery.
- **Meeting Attendance:** Team members consistently attended weekly supervisor meetings and internal discussions punctually, ensuring that communication was efficient and uninterrupted.

### **Professional Conduct**

- **Clear and Respectful Communication:** Communication platforms like MS Teams, Zoom, and email were used effectively to share updates, discuss progress, and address issues. Team discussions were conducted respectfully, fostering a collaborative atmosphere.
- **Mutual Support and Collaboration:** Team members supported each other through challenges, providing constructive feedback and assistance where needed. This collaborative environment enabled the team to overcome obstacles without delays.
- **Integration of Feedback:** Supervisor feedback was incorporated promptly and thoughtfully, improving the quality and functionality of the extension. The iterative feedback loop helped the team refine both technical and non-technical aspects of the project.

### **Outcome**

The professionalism demonstrated by the team throughout the project was reflected in its successful execution and timely delivery. The combination of thorough preparation, unwavering commitment, punctuality, and mutual respect ensured that every milestone was achieved with high standards. This professionalism not only contributed to the quality of the deliverables but also set a strong foundation for future collaborative projects.

## 13.0 CONCLUSION

The project represents a culmination of collaborative efforts, meticulous planning, and technical expertise, successfully addressing the critical need for enhanced online security. This extension not only meets the proposed objectives but also exceeds expectations in certain areas by incorporating additional functionalities that strengthen its usability and reliability.

The project adhered closely to its planned timeline, with over 90% of tasks completed on schedule and a final delivery date of **January 25, 2025**, as originally planned. Adjustments made during execution, such as refining testing phases and accommodating supervisor feedback, enhanced the overall quality of the deliverables without disrupting the schedule. **Key functionalities**, including Real-Time URL Monitoring, Phishing URL Detection, Basic Malware Detection, and Whitelist/Blacklist Management, were implemented successfully and optimized for performance. **Advanced features** such as Sandbox Integration for File Analysis and Machine Learning for Phishing Detection showcased the team's ability to innovate and expand the project's scope.

The extension strictly adhered to **ISO standards, GDPR, and Chrome Web Store Developer Policies**, ensuring compliance with international security and privacy guidelines. This focus on standards positions the extension as a robust and trustworthy tool for users, addressing modern cybersecurity challenges effectively. Through user-friendly features like categorized alerts, whitelist/blacklist management, and malware detection, the extension empowers users to browse the internet safely and confidently.

The project's success was also a testament to the team's professionalism, reflected in their preparedness, commitment, and punctuality throughout the development lifecycle. Collaborative tools such as MS Teams, Zoom, and email ensured seamless communication and progress tracking, while weekly supervisor meetings provided valuable guidance. The team's resilience in overcoming challenges such as API rate limitations and minor timeline adjustments further highlights their adaptability and problem-solving mindset.

While the project achieved significant milestones, challenges such as **Firefox compatibility issues** and delays in testing phases emphasized the importance of pre-development analysis and more accurate time estimates for task dependencies. These experiences provided valuable lessons that will inform future projects.

Looking forward, the extension's foundation enables the development of additional features, including full multi-browser compatibility, dynamic learning models, and enhanced reporting and analytics capabilities. These enhancements will broaden the extension's functionality and ensure it remains effective against evolving cybersecurity threats.

In conclusion, the Aegis Shield Phishing Detection Extension embodies a successful blend of innovation, collaboration, and adherence to best practices. Through thoughtful planning, proactive risk management, and technical excellence, the project achieved its goals while maintaining a focus on security, privacy, and user satisfaction. This achievement reflects the team's dedication and sets a strong precedent for future developments in the field of online security.

## 14.0 REFERENCES

- Belding, G. (2021, July 22). *ISO 27001 framework: What it is and how to comply* | Infosec. [Www.infosecinstitute.com. https://www.infosecinstitute.com/resources/management-compliance-auditing/iso-27001-framework-what-it-is-and-how-to-comply/](https://www.infosecinstitute.com/resources/management-compliance-auditing/iso-27001-framework-what-it-is-and-how-to-comply/)
- Best Practices and Guidelines*. (2024). Chrome for Developers. <https://developer.chrome.com/docs/webstore/program-policies/best-practices/>
- Boiral, O. (2011). Managing with ISO Systems: Lessons from Practice. *Long Range Planning*, 44(3), 197–220. <https://doi.org/10.1016/j.lrp.2010.12.003>
- Drozd, O. (2016). Privacy Pattern Catalogue: A Tool for Integrating Privacy Principles of ISO/IEC 29100 into the Software Development Process. *IFIP Advances in Information and Communication Technology*, 129–140. [https://doi.org/10.1007/978-3-319-41763-9\\_9](https://doi.org/10.1007/978-3-319-41763-9_9)
- General Data Protection Regulation*. (2016). <https://theanswerclub.com/wp-content/uploads/2024/04/General-Data-Protection-Regulation.pdf>
- ISO. (2024). *ISO - International Organization for Standardization*. Iso.org. <https://www.iso.org/home.html>
- ISO 27017: Comprehensive Guide to Cloud Security Standards*. (2024). Wwww.6clicks.com. <https://www.6clicks.com/resources/guides/iso-27017>