Phish Net
(Phishing Simulation Toolkit)

By:

Umar Waqar
27668

Ali Zaman Kayani
37539

Supervised by:
Dr. Jawaid Iqbal

Faculty of Computing
Riphah International University, Islamabad
Fall 2025

A Dissertation Submitted To:

Faculty of Computing,

Riphah International University, Islamabad

As a Partial Fulfilment of the Requirement for the Award of

the Degree of:

Bachelors of Science in Cyber Security

Faculty of Computing
Riphah International University, Islamabad

Final Approval

This is to certify that we have read the report submitted by ***Ali Zaman Kayani (37539) & Umar Waqar (27668)***, for the partial fulfilment of the requirements for the degree of the Bachelor of Science in Cyber Security (BS CYS). It is our judgment that this report is of sufficient standard to warrant its acceptance by Riphah International University, Islamabad for the degree of Bachelor of Science in Cyber Security (BS CYS).

Committee:

1

_____

Dr. Jawaid Iqbal
(Supervisor)

2

_____

Dr. Musharraf Ahmed
(HoD SE & CS)

Declaration

We hereby declare that this document "**Phish Net - Phishing Simulation Toolkit**" neither as a whole nor as a part has been copied out from any source. It is further declared that we have done this project with the accompanied report entirely on the basis of our personal efforts, under the proficient guidance of our teachers especially our supervisor **Dr. Jawaid Iqbal**. If any part of the system is proved to be copied out from any source or found to be reproduction of any project from anywhere else, we shall stand by the consequences.

Ali Zaman Kayani
37539

Umar Waqar
27668

Dedication

We dedicate our project, **Phish Net - Phishing Simulation Toolkit**, to the individuals who have consistently inspired and supported us along this journey. We truly dedicate our work to our parents and family. Their prayers, sacrifices, and love have always been our greatest assets. We have been able to overcome challenges and keep up our will to achieve our goals thanks to their assistance. We are also grateful to our teachers and instructors for being our inspirations and mentors. Their knowledge, understanding, and support have not only benefited our schooling but have also shaped the skills we need to do this task. Their confidence in us allowed us to take on this challenge. Lastly, we would want to express our gratitude to everyone who is working tirelessly to make the internet world safer. We think this project will help the cybersecurity community, which protects millions of individuals from unseen and unforeseen threats by the normal public. Furthermore, we are working towards a future where technological breakthroughs are accomplished without compromising security of the lay man.

Acknowledgement

First of all, we are obliged to Allah Almighty the Merciful, the Beneficent and the source of all Knowledge, for granting us the courage and knowledge to complete this Project.

We are grateful to the ***Faculty of Computing*** for creating a supportive learning atmosphere and to our seniors and classmates for their efforts, which improved the horizons of our experience. Last but not least, we are very thankful to our families for their unfailing love, support, and prayers, without them we would not have been able to reach this far.

Ali Zaman Kayani
37539

Umar Waqar
27668

Abstract

In today's world, the digital landscape is full of vulnerabilities, especially phishing attacks; that have evolved into one of the most pervasive and damaging cyber threats without a doubt, via exploiting the human vulnerabilities more than technological ones. Despite the increased investment in security infrastructure like SIEM and SOC solutions, organizations — particularly in regions like Pakistan — continue to suffer due to the lack of accessible, realistic, and localized training tools, guiding them in the ways to tackle these pervasive phishing threats. This project presents - **Phish Net: A Phishing Simulation Toolkit** - a scalable, customizable, and locally adaptable platform designed to **bridge the gap between awareness and action, reporting and training** in organizational cybersecurity. Thus, eliminating the long-suffered vulnerability of weak personnel.

Phish Net empowers organizations to **simulate real-world phishing attacks** in a controlled sandbox environment, monitor each and every users' interactions, and generate detailed and thorough analytics to measure people awareness and behavioural risk. Unlike existing international solutions that are often prohibitively expensive, rigid, or disconnected from regional threats, Phish Net is **tailored for local needs**, offering **dynamic phishing campaign management**, **role-based access control**, and **real-time behavioural insights** — all within a user-friendly, bespoke and secure solution.

Built using a modern tech stack comprising **React, Django, PostgreSQL API integration**, the toolkit ensures robust performance, modular scalability, and seamless automation of phishing simulations. Through our intuitive dashboards, granular reporting, and API-driven customization, Phish Net not only enhances cybersecurity awareness programs but also enables organizations to proactively adapt their training strategies based on the people's response patterns and user behavioural analytics.

Table of Content

List of Figures

List of Tables

Chapter 1: Introduction

Chapter 1: Introduction

The rising tide of cyber threats, particularly phishing attacks, has highlighted a critical gap in organizational security: human awareness and preparedness. Despite technological advancements, phishing remains effective due to its social engineering tactics, which exploit user behaviour rather than system vulnerabilities. For humans are the weakest link in any technological infrastructure – susceptible to emotions and trickery. This project introduces **Phish Net**, an advanced phishing simulation toolkit developed to address this gap. Phish Net aims to provide organizations, especially within Pakistan, with an interactive and locally adaptable solution to train employees, evaluate susceptibility, and promote cybersecurity awareness through real-world simulations.

1.1 Opportunity & Stakeholders

The rapidly increasing frequency and sophistication of phishing attacks present a troublesome environment and a perfect opportunity to develop a home-grown, cost-effective testing and simulation solution for our country's organizations. Global platforms such as KnowBe4 and Cofense offer phishing simulations but often fall short in terms of affordability, scalability, flexibility, and regional relevance. Phish Net capitalizes on this market gap by providing a customizable platform tailored for small to medium scale enterprises in Pakistan. With the ability to be tailored to everyone's desires, whether it be public organizations dealing with critical data or private institutions having to entertain global benchmarks and safeguard sensitive client/user data.

1.1.1 Primary Stakeholders

- **Organization's IT Departments:** Implement and manage phishing simulations.
- **People:** Primary participants and target group for awareness training.
- **Cybersecurity Consultants:** Utilize simulation data to improve training programs and develop policies.

1.2 Motivation & Challenges

The project is driven by the urgent need to equip organizations with tools that go beyond the book-taught typical conventional cybersecurity awareness techniques. Many businesses, especially in developing regions, lack access to affordable training solutions or the reliability of their access. Additionally, phishing tactics are becoming more deceptive and difficult to detect, necessitating meticulously developed proactive simulation-based approaches. All while remaining in an enclosed, safe and secure sand box environment, so nothing goes out of hand.

1.2.1 Key Challenges

- Ensuring **email deliverability** while avoiding spam filters.
- Developing **user-friendly yet robust interfaces** for campaign and analytics management.
- Maintaining **system security** against misuse of the platform.
- Ensuring access controls and privilege integrity.
- Handling **scalability** across multiple organizations and user roles.

1.3 Significance of study

This study adds up significantly to the cybersecurity industry specifically in the awareness and training field. By developing Phish Net, we'll be able to addresses critical real-world challenges associated with phishing attacks and offers a sustainable, scalable solution for the local landscape. It also provides the framework for evaluating the organization's human behaviour in response to cyber threats. Thus, adding to the creation of more robust security policies and training modules.

1.3.1 Practical Implications

- Enhances employee vigilance and threat detection.
- Supports IT teams in tailoring security interventions.
- Contributes to national cybersecurity resilience.
- Assistance in improving the social engineering calibre of the lay-man.
- Granting compliance with international standards and practices.

1.4 Goals & Objectives

The primary goal of this project is to develop a fully functional, modular and a secure phishing simulator. Such a system will allow our nation's organizations conduct realistic phishing campaigns, monitor the user engagement, and generate actionable reports.

1.4.1 Project Objectives

- Design and develop a **React-Django** based phishing simulator.
- Integrate **API** for dynamic email delivery and tracking.
- Provide a **user-friendly dashboard** with campaign management tools.
- Enable **detailed analytics** for behavioural insight.
- Implement **role-based access control** for multi-user environments.

1.5 Scope of project

Phish Net focuses on the development of a phishing simulation platform with a firm emphasis on localization, modularity, customization, and above all the real-time reporting. It is designed for deployment in both self-hosted and cloud-based environments, to better facilitate everyone's implementation preference. The toolkit supports multiple custom email templates, and campaign genres, with a backend database highly capable of scaling as the user load and the company requirements expand. While having a robust back bone in the form of a reliable database storing all this data into segregated entities to be utilized by the system. Without the fear of conflict between the two.

1.5.1 In-Scope Features

- User authentication and RBAC
- Phishing campaign creation and automation
- Tracking of email opens, clicks, and submissions
- Behavioural analytics and reporting

1.6 Summary

The project - Phish Net addresses a crucial vulnerability in our national posture by eliminating the threat of pervasive phishing attacks. Satisfying the nation's need in organizational cybersecurity via training by simulating phishing attacks to raise awareness and monitor behavioural responses, to help organizations improve their security posture through rigorous testing and training campaigns. Through its modular design, localized relevance, and technical robustness, the toolkit offers a much-needed alternative to existing international platforms. The project sets the stage for future enhancements if the time permits for introducing AI-driven campaigns, gamification, and integration with broader cybersecurity systems.

1.6.1 Conclusion of Chapter

This chapter introduced the background, opportunities, and motivations behind the development of Phish Net. It also outlined the project's goals, scope, and the significance of its contribution to cybersecurity awareness and training, and to the overall community, for welfare purposes.

Chapter 2:
Market Survey

Chapter 2: Market Survey

The effectiveness of a cybersecurity training solution is not solely determined by its technical features but also by how well it addresses existing market needs. This chapter provides an overview of existing phishing simulation products, analyses their features, and highlights the gaps that Phish Net aims to fill. It also contextualizes the need for a localized and adaptable toolkit through a comparative market study.

2.1 Introduction

Phishing continues to be a dominant method used in cyberattacks worldwide. For it is the root cause of every cyber dilemma, as social engineering the opening shot of every cyber offense. In response, several platforms have emerged offering phishing simulation and training services. However, many of these platforms cater to international markets and fail to address the unique requirements of organizations in developing countries like Pakistan. This chapter explores the current market landscape and outlines how Phish Net differentiates itself.

Table 2.1: Comparison of Existing Phishing Awareness Platforms

Platform	Open Source	Campaign Management	Email Tracking	Landing Pages	Reporting	Training Modules	Price Range
Gophish	✅ Yes	✅ Yes	✅ Yes	✅ Yes	Basic	❌ No	Free
KnowBe4	❌ No	✅ Yes	✅ Yes	✅ Yes	Advanced	✅ Yes	$$$$
Lucy Security	❌ No	✅ Yes	✅ Yes	✅ Yes	Advanced	✅ Yes	$$$
PhishNet	✅ Yes	✅ Yes	✅ Yes	✅ Yes	Advanced	✅ Yes	Free

Table 2.1 presents a feature-wise comparison of major phishing awareness and simulation platforms.

2.1.1 Importance of Market Research

Conducting a market survey provides vital insight into user needs, competitive offerings, pricing models, and technological trends. It serves as the foundation for designing a product that is both innovative and relevant to its target audience. While also helping the developers note the comparative strengths and weaknesses of every product if put head-on. Not only that it also helps distinguish the gaps for granting the need to produce a more refined and further enhanced product. For such research helps in understanding the current market trends as well.

2.2 Products Overview

There are several well-known phishing simulation platforms currently dominating the global cybersecurity training market. Notable among them are:

- **KnowBe4** – Offers security awareness training and simulated phishing attacks. Known for a vast template library and analytics.
- **Cofense (formerly PhishMe)** – Focuses on phishing threat management and user behaviour tracking.
- **Barracuda PhishLine** – Integrates security awareness with customizable phishing attack simulations.

Table 2.2: Justification of Selected Technologies

Component	PhishNet	Alternatives	Justification
Frontend	React 18 + Vite	Vue.js, Angular	Fast development, modern tooling, active community
Backend	Node.js + Express	Python Flask, Django	JavaScript full-stack, async I/O, large npm ecosystem
Database	PostgreSQL 15	MySQL, MongoDB	ACID compliant, JSON

Table 2.2 shows the selection of technologies for **PhishNet** across all layers of the system

2.2.1 Limitations of Existing Products

While these platforms are robust, they present limitations:

- High costs and subscription fees.
- Lack of localization for our regional contexts.
- Little to no template customization.
- Dependence on foreign cloud infrastructure
- Data privacy concerns & lack of customization

2.3 Comparative Analysis

For a brief comparison following table could be viewed:

Table 2.3 – Brief Comparison Analysis

Features	**KnowBe4**	**Cofense**	**Proofpoint**	**Phish Net**
Local Language Support	✗	✗	✗	✓
Custom Template Creation	✓	✓	✗	✓
Behavioural Analytics	✓	✓	✗	✓
API Integration	✓	✗	✓	✓
Deployment Options	Cloud	Cloud	Self-hosted	Cloud / Self-hosted
Cost Efficiency	✗	✗	✓	✓

Table 2.3 justifies the selection of **PhishNet** across all genres of the system.

For a more in-depth Comparison kindly view table 2.4:

Table 2.4 – Detailed Comparison Analysis

Feature	Proofpoint	Infosec	KnowBe4	SafeTitan	Cofense	Phish Net
Email Protection	✓					✓
Advanced BEC Defense	✓					✓
Sandboxing	✓					✓
Click-Time Protection	✓					✓
Email Continuity	✓					✓
Security Awareness Training	✓	✓	✓	✓	✓	✓
Phishing Simulations	✓	✓	✓	✓	✓	✓
Recommendations			✓			✓
SmartRisk™ Agent			✓			✓
Localized Content			✓			✓
Advanced Reporting	✓	✓	✓	✓	✓	✓
Gamification		✓	✓			✓
Custom Test Building		✓	✓			✓
Non-Email Based Testing		✓	✓			✓
Pre-Assessments		✓	✓			✓
Prebuilt Training Library		✓	✓			✓
Integration with Security Tools	✓		✓		✓	✓
Advanced Reporting	✓	✓	✓	✓	✓	✓

2.3.1 Key Differentiators of Phish Net

- Focus on **localization** and cultural relevance.
- **Dual deployment options** (cloud and self-hosted).
- **Advanced analytics** for user behaviour assessment.
- Encouragement to information collaboration
- **Affordable** and accessible for SMEs in Pakistan.

2.4 Research Gaps

Despite the availability of global platforms, the following gaps persist in the current solutions:

- Absence of regional threat database integration.
- Inaccessibility to small businesses due to cost.
- Inflexibility in customizing attack scenarios.
- Lack of system customization
- Lack of behavioural training feedback based on regional social engineering tactics.

2.4.1 Gap in Cybersecurity Training Tools

Current tools fail to combine localization, affordability, and behaviourally driven analytics in a single, scalable solution suitable for developing countries.

2.5 Problem Statement

Organizations in Pakistan lack access to a secure, affordable, and locally adaptable phishing simulation platform that can:

- Mimic real-world phishing attacks relevant to the regional context.
- Offer customizable templates and dynamic campaign options.
- Generate actionable behavioural insights through detailed analytics.
- Integrate easily with existing IT infrastructure and security protocols.

2.5.1 Core Problem

Without an accessible and tailored solution like Phish Net, organizations remain vulnerable to phishing threats due to inadequate employee training and awareness. While facing lack of control and customization in their operations.

2.6 Summary

This chapter presented a detailed market analysis highlighting the limitations of existing phishing simulation products and the unique value proposition being offered by Phish Net to our regional market. While having a close concern of their major interests to best fit their needs and fulfill their desires. By focusing on localization, affordability, customization and comprehensive analytics, Phish Net aims to bridge a critical gap in cybersecurity training for organizations in Pakistan.

Chapter 3:
Requirements & System Design

Chapter 3: Requirements & System Design

This chapter details the functional and non-functional requirements, architectural blueprint, system components, and threat modelling strategies essential to the secure and efficient development of Phish Net.

3.1 Introduction

The increasing frequency and sophistication of phishing attacks pose a significant threat to digital infrastructures. To address this, the Phish Net Phishing Simulator Toolkit is designed as an advanced, customizable, and secure phishing simulation platform tailored for both local and global organizational needs.

PhishNet operates within a complex organizational IT ecosystem, interfacing with multiple external systems to deliver phishing awareness training. The platform integrates with corporate email gateways, identity providers, and DNS infrastructure while maintaining clear security boundaries between internal components and external services.

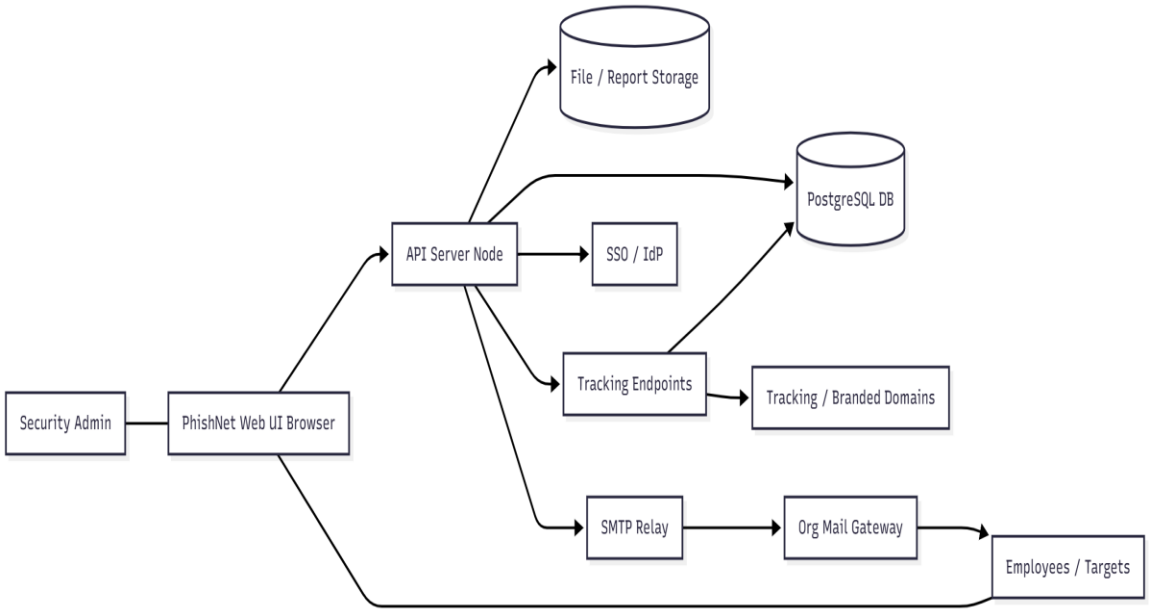


Figure 3.1: Infrastructure Components

Figure 3.1 illustrates the system context, showing how PhishNet interacts with administrators, target employees, and external infrastructure components.

Table 3.1: Database Tables Overview

Table Name	Primary Key	Foreign Keys	Description	Record Estimate
users	id	-	System administrators and analysts	~50
organizations	id	-	Onboarded companies	~10
targets	id	organization_id	Employees targeted by simulations	~5,000
campaigns	id	owner_user_id, template_id	Phishing simulation campaigns	~500
campaign_results	id	campaign_id, target_id	Tracks user interactions	~50,000
templates	id	created_by	Email/landing page templates	~100
trainings	id	created_by	Awareness training content	~50
user_training	id	user_id, training_id	Training completion tracking	~10,000
sessions	sid	user_id	Active session management	~200

Table 3.1 summarises the database tables used in PhishNet, showing relationships, foreign keys, and estimated record counts based on system scale.

3.2 System Architecture

Phish Net employs a modular client-server architecture. The front end, developed in React with Vite and Tailwind CSS, communicates with the Django backend via RESTful APIs. Creating a customizable template engine. PostgreSQL is used as the relational database, ensuring robust and scalable data handling. JWT and OAuth secure authentication mechanisms, supplemented by Google reCAPTCHA and MFA, safeguard access control. While constructing an automated mailing and notification mechanism.

PhishNet follows a modern multi-tier architecture that separates presentation logic, business logic, and data persistence. The system is composed of several specialized containers, each handling distinct responsibilities within the overall platform.

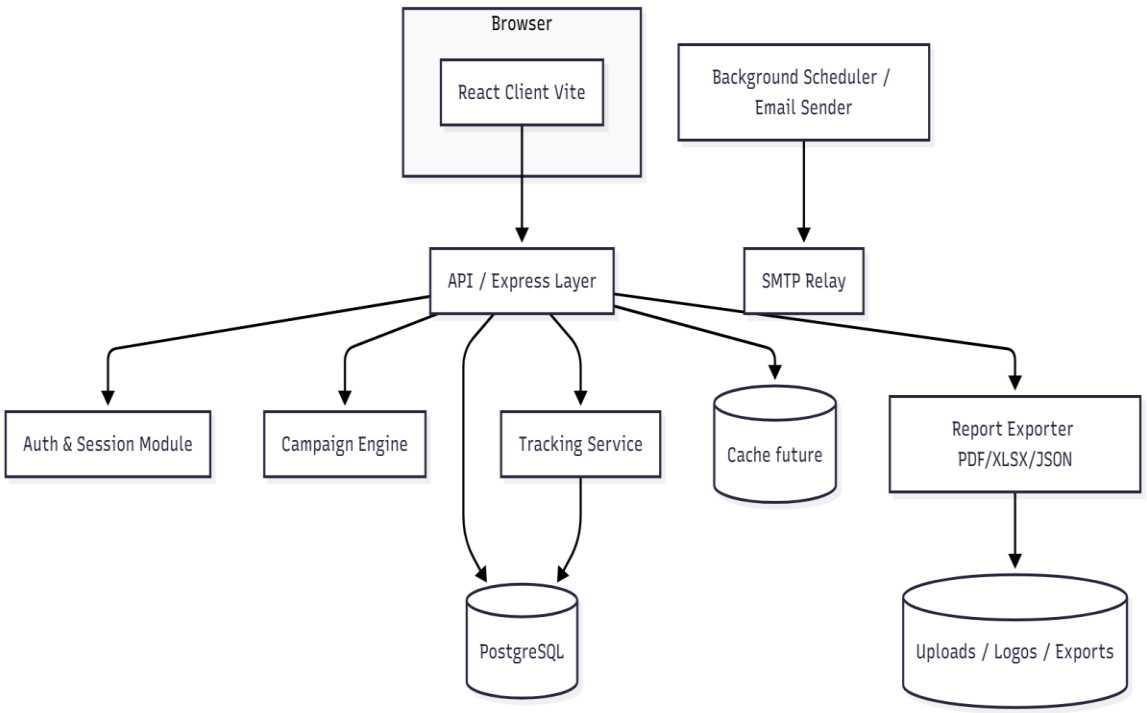


Figure 3.2: Container – Level Architecture

Figure 3.2 presents the container-level architecture, detailing the internal structure of the PhishNet application. The frontend React client communicates with a centralized Express API layer, which orchestrates interactions across specialized backend services including campaign management, tracking, reporting, and background job processing.

Table 3.2: API Endpoints Summary

Endpoint	Method	Auth Required	Description	Response Format
/api/auth/login	POST	❌ No	Authenticate user	{ token, user }
/api/campaigns	GET	✅ Yes	List all campaigns	{ campaigns: [] }
/api/campaigns	POST	✅ Yes	Create new campaign	{ id, name, status }
/api/campaigns/:id	GET	✅ Yes	Get campaign details	{ campaign: {} }
/api/campaigns/:id/launch	POST	✅ Yes	Launch campaign	{ success: true }
/api/reports/export	POST	✅ Yes	Export report	{ downloadUrl }
/api/targets	POST	✅ Yes	Import target list	{ imported: 50 }
/api/trainings/:id/enroll	POST	✅ Yes	Assign training	{ enrolled: true }
/track/open/:id	GET	❌ No	Track email open	1×1 pixel image
/track/click/:id	GET	❌ No	Track link click	HTTP 302 redirect

Table 3.2 documents key RESTful API endpoints exposed by PhishNet, their access control requirements, and expected responses.

3.3 Functional Requirements

Functional requirements are the actual work that is to be produced by the project when put into operation. They are the core operations of the Phish Net system that directly serve user and system-level functionalities.

The PhishNet data model follows a normalized relational design optimized for tracking campaign execution and user behaviour. The schema supports multi-tenancy, campaign management, user training, and comprehensive analytics.

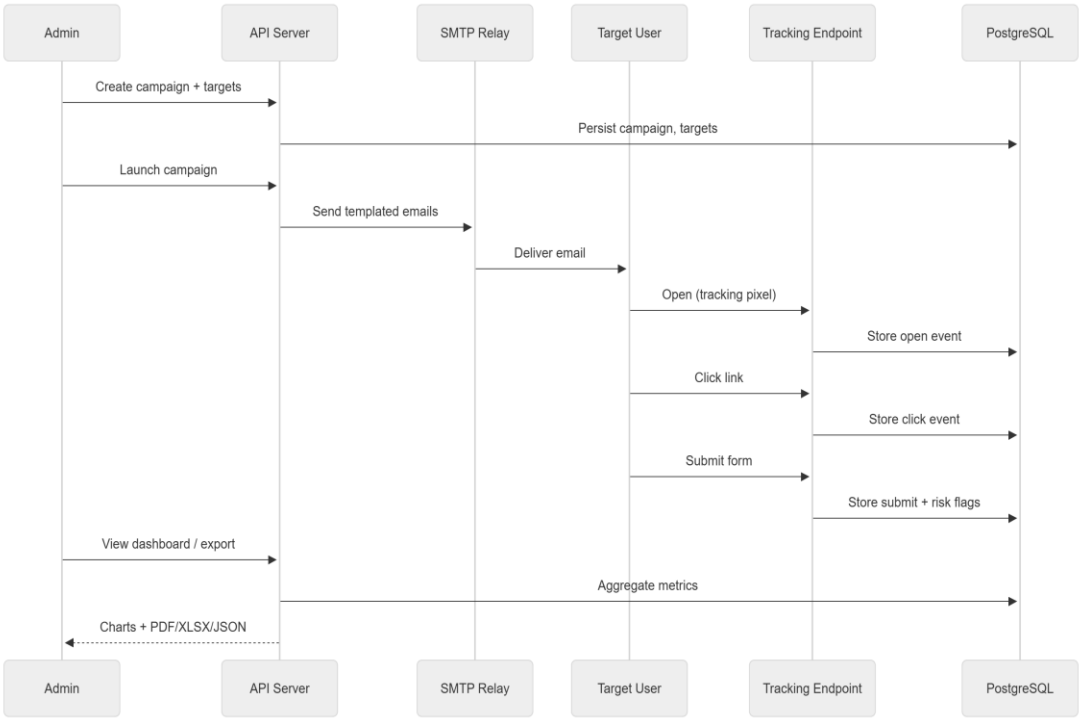


Figure 3.3: Entities & Relationships

Figure 3.3 illustrates the core entities and their relationships within the PostgreSQL database. The model centres around the CAMPAIGNS and CAMPAIGN_RESULTS tables, which capture the lifecycle of phishing simulations and detailed interaction metrics for each target.

Following are the main needs that this project is asked to fulfil:

- User Registration and Authentication (RBAC)
- Campaign creation and email template management
- Email scheduling (batch, burst, normal modes)
- Real-time tracking of email interactions (open, click, submit)
- Organization and department management
- Custom phishing scenario deployment
- Reporting and analytics dashboard
- API integration for automation and SIEM compatibility

Table 3.3: User Roles & Permissions (RBAC)

This Role-Based Access Control (RBAC) matrix defines what each user type can view or perform within PhishNet.

Role	View Dashboard	Create Campaigns	Launch Campaigns	Manage Users	Export Reports	View Training	Assign Training
Admin	✅	✅	✅	✅	✅	✅	✅
Analyst	✅	✅	✅	❌	✅	✅	✅
Viewer	✅	❌	❌	❌	✅	✅	❌
Employee	⚠ Limited	❌	❌	❌	❌	✅	❌

3.4 Non-functional Requirements

The non-functional requirements are the that this project will help to achieve on completion. They ensure the overall quality, security, and maintainability of the system in diverse operating environments.

Following are the non-functional requirements:

- Scalability: Should handle multiple organizations and user groups concurrently
- Security: Must protect against unauthorized access and data breaches
- Usability: Intuitive UI/UX for non-technical users
- Performance: Real-time response tracking and minimal latency

Maintainability:

- Modular codebase for easy updates
- Availability: High uptime and reliable hosting

3.5 Design Diagrams

The design diagrams include the following provided in the Appendix A:

- Use Case Diagrams: Illustrating user interactions with the system
- Class Diagrams: Mapping core system entities and relationships
- Sequence Diagrams: Detailing interactions in phishing campaign lifecycle
- Deployment Diagrams: Showcasing containerization and server allocation

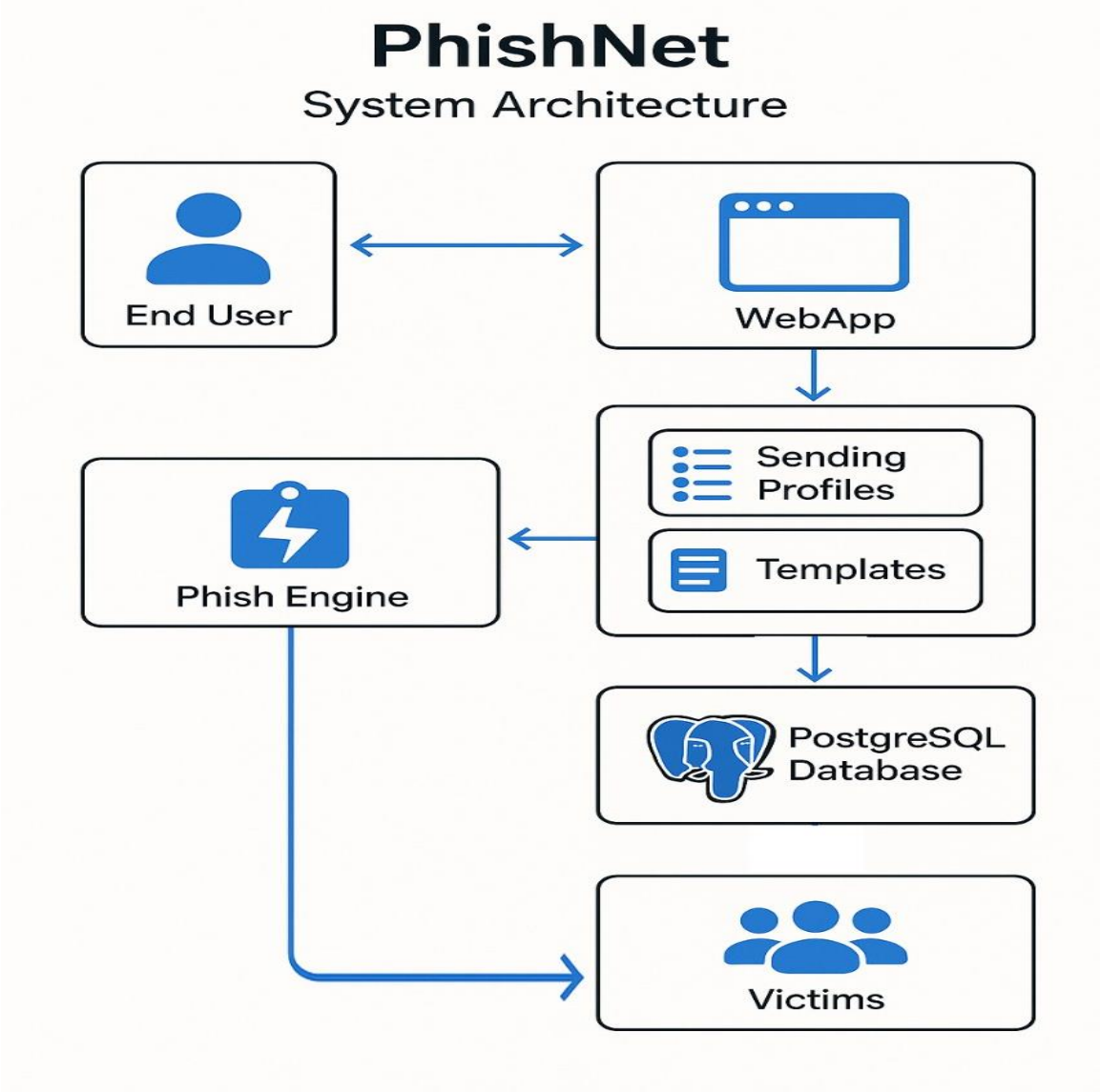


Figure 3.4: Design Diagram

Figure 3.4 gives an overview of the system design and its key components. Explaining the key functionalities of the system as well.

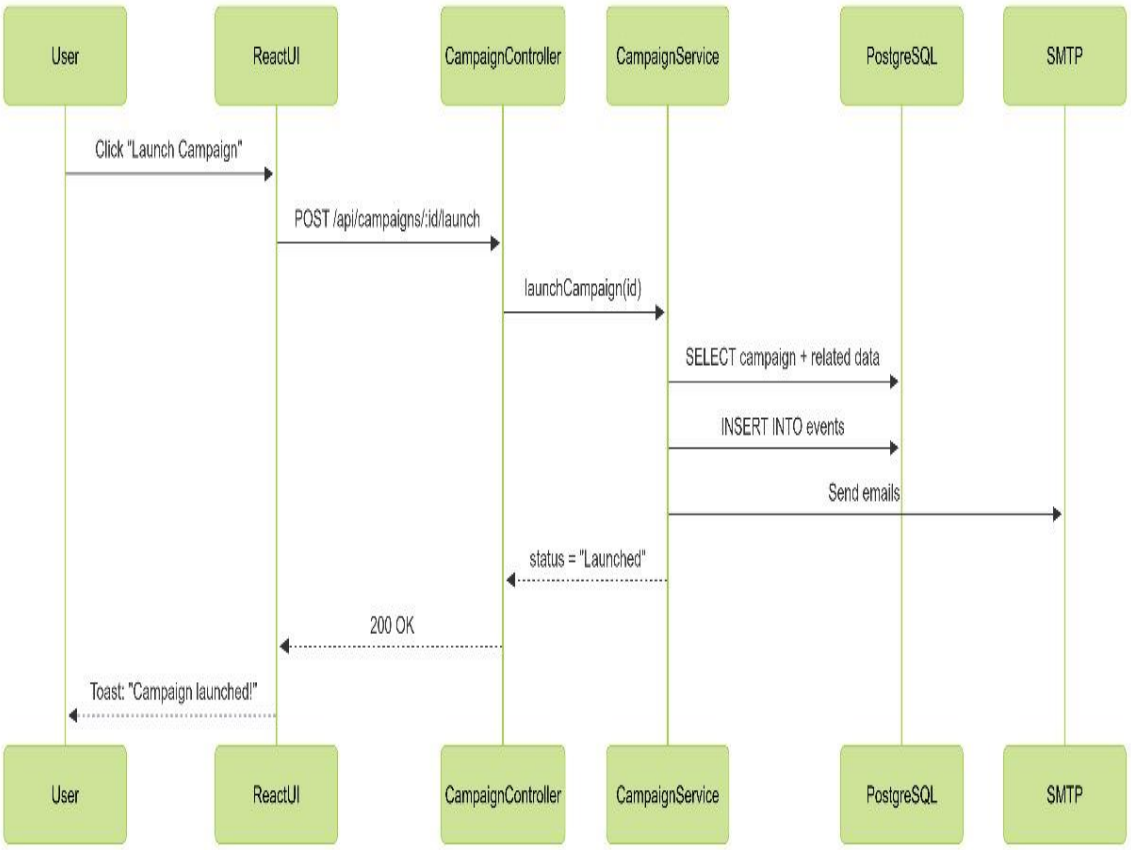


Figure 3.5: Sequence Diagram

Figure 3.5 gives an overview of the system components and its key interdependencies. Explaining the flow of the system as well.

3.6 Hardware & Software Requirements

Following are thew hardware and software intakes that are required by the project:

Hardware:

- Minimum: 8 GB RAM, i5 Processor, 250 GB SSD
- Recommended: 16 GB RAM, i7 Processor, 500 GB SSD

Software:

- Frontend: React, Tailwind CSS, Vite
- Backend: Django (Python), Database
- Database: PostgreSQL
- Tools: Docker, GitHub, GitHub Actions, Jest, Cypress

3.7 Threat Scenarios

Following are the threat scenarios that are to be produced/achieved in the project:

- Email spoofing detection bypass
- Phishing link redirection to malicious domains
- Credential harvesting through cloned pages
- Social engineering exploitation
- Misuse of toolkit by unauthorized users

3.8 Threat Modelling Techniques

Phish Net utilizes a structured and proactive approach to threat modelling based on the STRIDE framework. This helps in identifying potential security threats and implementing appropriate countermeasures throughout the development lifecycle.

Following are the most widely recognized types:

1. STRIDE Model (Microsoft Model)

- **Purpose:** Developed by Microsoft, this model categorizes threats into six distinct types to ensure a holistic assessment.
- **Acronym Breakdown:**
  - **S – Spoofing Identity:** Impersonating a user or system (e.g., fake login pages in phishing).
  - **T – Tampering:** Unauthorized modification of data or code.
  - **R – Repudiation:** Users denying actions (e.g., claiming they didn't click a phishing link).
  - **I – Information Disclosure:** Accidental or malicious data leakage.
  - **D – Denial of Service (DoS):** Disrupting service availability.
  - **E – Elevation of Privilege:** Gaining higher access rights than permitted.
- **Application to Phish Net:** STRIDE helps identify vulnerabilities in authentication, privilege escalation, and spoofed phishing templates.

2. DREAD Model

- **Purpose:** Quantifies and prioritizes threats based on their potential impact and likelihood.
- **Acronym Breakdown:**
  - **D – Damage Potential**
  - **R – Reproducibility**
  - **E – Exploitability**
  - **A – Affected Users**
  - **D – Discoverability**
- **Usage:** Each factor is rated on a scale (e.g., 1–10) to produce a numerical risk score.

3. PASTA (Process for Attack Simulation and Threat Analysis)

- **Purpose:** A risk-centric methodology emphasizing attacker perspective and business impact.
- **Steps:** Includes seven stages — from defining objectives and identifying threats to conducting simulations and setting mitigation strategies.

4. LINDDUN Model

- **Purpose:** Privacy-focused threat modelling approach — identifies threats to user data confidentiality and compliance.
- **Acronym Breakdown:**
  - **L – Linkability**
  - **I – Identifiability**
  - **N – Non-repudiation**
  - **D – Detectability**
  - **D – Disclosure of Information**
  - **U – Unawareness**
  - **N – Non-compliance**

5. Attack Trees

- **Purpose:** Visual, hierarchical representation of how an attack might occur.
- **Structure:** The root node represents the attacker's goal (e.g., compromise campaign integrity), and branches represent various paths or exploits to achieve it.
- **Application to Phish Net:** Helps visualize different ways the platform or users could be compromised — e.g., unauthorized admin access, campaign data exfiltration, or spoofed notifications.

6. OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation)

- **Purpose:** A self-directed, organization-level methodology emphasizing asset-based risk management.
- **Phases:** Identify assets, identify threats, and develop mitigation strategies.

7. VAST (Visual, Agile, and Simple Threat Modelling)

- **Purpose:** Designed for large-scale, agile environments to integrate threat modelling with DevOps and CI/CD pipelines.
- **Characteristics:**
    - Focuses on scalability and automation.
    - Produces visual, actionable insights.

Table 3.8 – Threat Modelling Summary Table

Technique	Focus	Best For	Phish Net Relevance
STRIDE	System-based threat identification	Architecture design phase	Authentication, spoofing prevention
DREAD	Quantitative risk scoring	Risk prioritization	Ranking phishing simulation vulnerabilities
PASTA	Attack simulation & impact	Advanced threat analysis	Attacker behaviour simulation
LINDDUN	Privacy protection	Data compliance	User data & analytics privacy
Attack Trees	Visual representation	Threat visualization	Understanding attack paths
OCTAVE	Organizational risk	Asset management	Policy & control mapping
VAST	DevOps integration	Scalable environments	Continuous security testing

Table 3.8 shows a summary of the threat modelling techniques used in the project life cycle.

Whereas, Phish Net utilizes the STRIDE model:

- Spoofing: Prevented through domain verification and DMARC
- Tampering: Mitigated by secure API validation and encryption
- Repudiation: Audit logs for user activity
- Information Disclosure: Encrypted data storage
- Denial of Service: Rate limiting and failover mechanisms
- Elevation of Privilege: Enforced RBAC and MFA

3.9 Threat Resistance Model

Phish Net's security is built on a multi-layered threat resistance architecture that integrates defensive mechanisms across all components of the system. This includes preventive, detective, and responsive controls to safeguard against known and emerging cyber threats.

Phish Net's security is built on layered defences:

- Application Layer: Input sanitization, token-based authentication
- Network Layer: TLS encryption, firewall rules
- Data Layer: Role-based DB access, hashed credentials
- Monitoring: Logging, intrusion detection, anomaly alerts

In a nutshell, this chapter detailed the architectural vision, system modules, and security strategies that guide the implementation of Phish Net. It lays a strong foundation for a secure, scalable, and user-friendly phishing simulation platform, ready to be integrated into any organization's cybersecurity awareness framework.

Chapter 4:
Proposed Solution

Chapter 4: Proposed Solution

4.1 Introduction

Phish Net addresses a critical gap in cybersecurity awareness by offering a localized and customizable phishing simulation toolkit. As the phishing attacks grow in frequency and complexity, and imagination so does the need for an adaptive, secure, and accessible training solution has become critically paramount. That is not only robust but tailored to everyone's use case. This chapter presents the comprehensive design of the proposed solution, covering the conceptual model, data lifecycle, tools, and evaluation strategies.

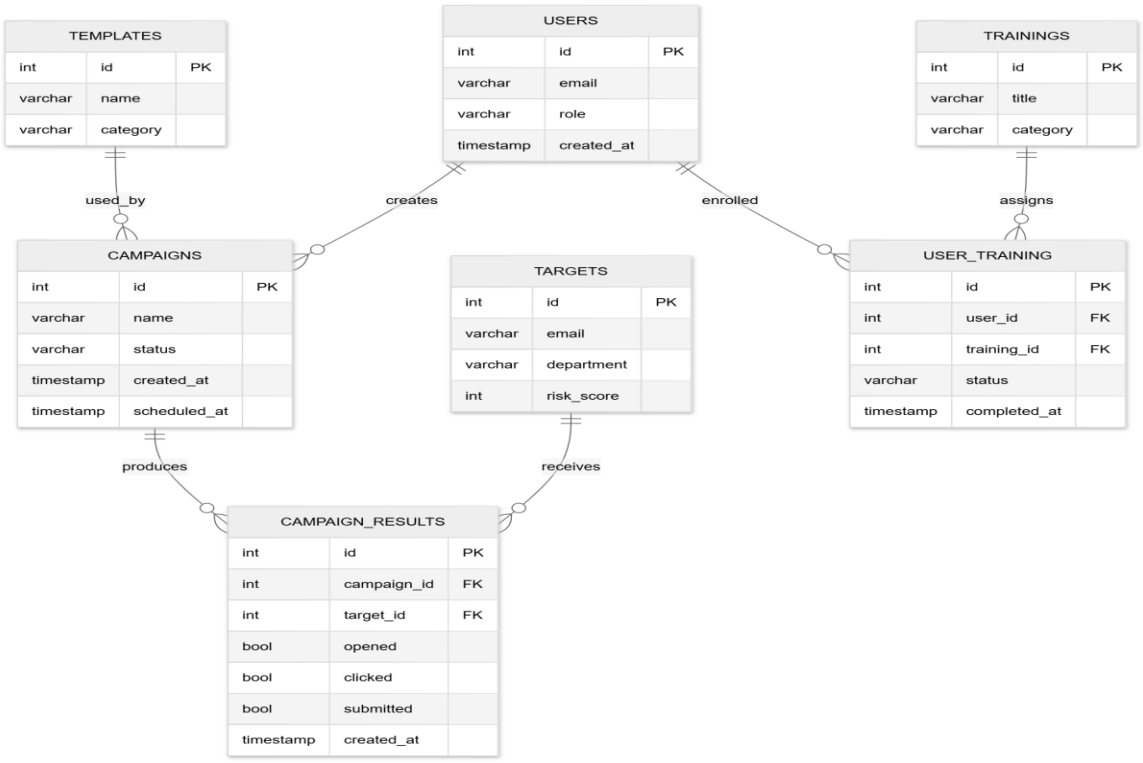


Figure 4.1: Phishing Campaign Lifecycle

Figure 4.1 is illustrating the complete lifecycle of a phishing campaign. The flow administrator launching campaign, email delivery, user interaction tracking - metrics.

Table 4.1: Campaign Template Categories

Category	Template Count	Difficulty	Avg Click Rate	Avg Submit Rate	Example Technique
Email Spoofing	15	Medium	35%	12%	Fake IT password reset
Spear Phishing	12	High	28%	18%	Personalized CEO fraud
Pretexting	10	High	42%	22%	Fake HR survey
Baiting	8	Low	55%	8%	Free gift card link
Vishing/Smishing	5	Medium	30%	15%	SMS attack link
Watering Hole	3	Low	20%	5%	Compromised legit site

Table 4.1 lists the phishing simulation template categories with average engagement metrics to illustrate realism and challenge balance.

4.2 Proposed Model

The proposed model for Phish Net integrates modularity, customization, automation, and scalability. It consists of a multi-tiered architecture including a React-based front end, Django backend, and constructing the customizable campaign engine. RESTful APIs connect these components, while PostgreSQL serves as the primary database. The model supports user-defined phishing scenarios, RBAC, scheduling modes, and real-time analytics. Security is enforced via OAuth, JWT, MFA, and Google reCAPTCHA.

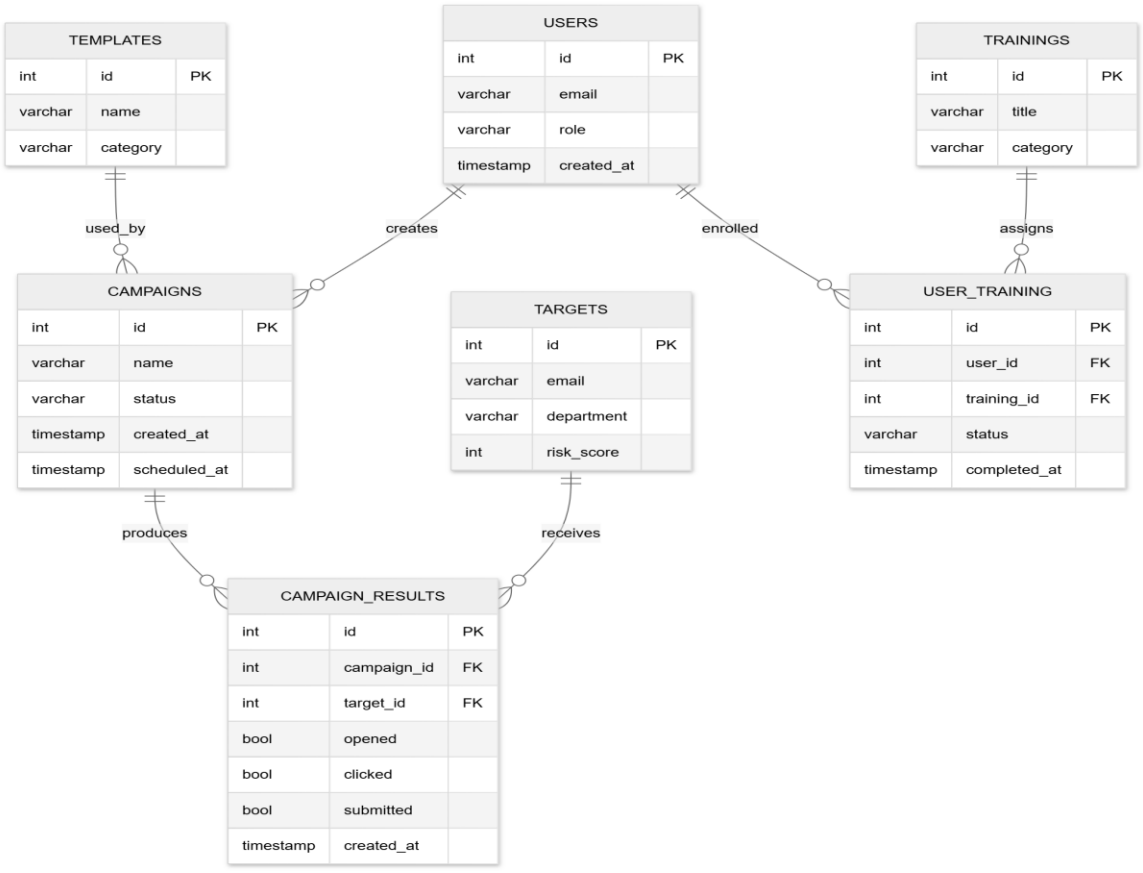


Figure 4.2: Phishing Lifecycle State Machine

Figure 4.2: Illustrating campaign lifecycle state machine, showing states and the events trigger transitions. Enables administrators pause campaigns mid-execution for review, resume, and archive completed campaigns.

Table 4.2: Report Export Formats Comparison

Format	File Extension	Use Case	Features	Theme Support	Avg Size
PDF	.pdf	Executive summaries	Charts, branding	✅ Light/Dark	2–5 MB
Excel	.xlsx	Data analysis	Pivot tables, formulas	✅ Themed headers	0.5–2 MB
JSON	.json	API integration	Structured output	❌ N/A	100–500 KB
CSV	.csv	Spreadsheet import	Simple, lightweight	❌ N/A	50–200 KB

Table 4.2 compares file formats supported by PhishNet's report generator, outlining use cases and typical file sizes.

4.3 Data Collection

Phish Net simulates the phishing campaigns using customizable templates and deployment strategies. Data points collected include email open rates, link clicks (if any), credential submissions (type of submission), cursor movements, email reporting rate and user response times. This data is anonymized and stored securely via hashing to analyse the effectiveness of simulations across various departments and organizations.

4.4 Data Pre-Processing

Pre-processing ensures data integrity and usability. This includes deduplication, timestamp normalization, and user segmentation. Email metadata is extracted for trend analysis, while interaction logs are filtered and structured for real-time dashboard updates. All sensitive data is hashed before processing.

4.5 Tools & Techniques

Phish Net employs a robust set of tools to ensure performance and security. The marvel of a tech stack includes React, Tailwind CSS, Vite for the frontend, and Django. Docker containers are used for efficient deployment, ensuring isolation and scalability. CI/CD is managed through GitHub Actions, while testing is done using Jest and Cypress.

4.6 Evaluation Metrices

To gauge effectiveness, Phish Net uses various evaluation matrices:

- *Click-through rate (CTR):* Measures engagement with phishing emails
- *Submission rate:* Tracks how many users enter credentials
- *Time-to-click:* Indicates user reflex and awareness
- *Training effectiveness:* Pre- and post-campaign comparison
- *System performance:* Latency, uptime, and resource usage

4.7 Chapter Summary

This chapter was curated for the sole purpose of granting an insight into the project – “Phish Net”, the phishing simulation toolkit. Focusing on its system designing, data strategy and the implementation tools. While discussing the evaluation methods and metrices. In a nutshell, these components ensure Phish Net is a well secure, scalable, and a highly effective training and simulation-based solution for combatting the rapidly escalating and shifting phishing dilemmas that are found frequently in the real-world corporate landscape.

Chapter 5:
Implementation & Testing

Chapter 5: Implementation & Testing

5.1 Security Properties Testing

Phish Net has been designed with security as an apex priority. Security properties such as authentication integrity, confidentiality of stored data, secure API transactions, role-based access control (RBAC), and multi-factor authentication (MFA) were meticulously implemented while rigorously tested. These tests ensured the platform could withstand the usual cybersecurity threats like SQL injection, cross-site scripting (XSS), and brute-force attacks. Automated security scanners like OWASP ZAP and manual code reviews were used to validate compliance with secure coding practices. Along with tools like Jest and Cypress for ensuring code integrity.

Table 5.1: Functional Test Cases Summary

Test ID	Module	Test Case	Input	Expected Output	Status
TC-001	Auth	Valid login	admin@test.com, password123	Redirect to dashboard	✅ Pass
TC-002	Auth	Invalid password	admin@test.com, wrong pass	Error message	✅ Pass
TC-003	Campaigns	Create draft	Campaign name, template ID	Status “draft”	✅ Pass
TC-004	Campaigns	Launch campaign	Campaign ID	Status “running”	✅ Pass
TC-005	Tracking	Email open	Tracking pixel URL	Open logged	✅ Pass
TC-006	Tracking	Link click	Redirect URL	Click recorded	✅ Pass
TC-007	Reports	Export PDF	Campaign ID	File downloaded	✅ Pass
TC-008	Training	Assign module	User + training IDs	Record created	✅ Pass
TC-009	RBAC	Viewer creates campaign	API request	403 Forbidden	✅ Pass

Table 5.1 documents core functional test cases, inputs, expected results, and current pass/fail status.

5.2 System Setup

The deployment of Phish Net follows a containerized approach using Docker, ensuring consistency across development, testing, and production environments, while being the perfect route for deployment. PostgreSQL database instances, Django backend servers, and React-based frontend containers are orchestrated using Docker Compose. Environment variables are securely managed, and SSL certificates are integrated to ensure secure HTTPS connections.

Table 5.2: Performance Benchmarks

Operation	Load Scenario	Avg Response	Throughput	DB Queries	Status
Dashboard load	100 users	1.2 s	83 req/s	8	✅ Good
Campaign creation	50 users	450 ms	111 req/s	3	✅ Good
Email sending	1000 emails	30 s	33 emails/s	1000	⚠️ Acceptable
Report export	10 exports	8.5 s	1.2 req/s	50	⚠️ Acceptable
Tracking pixel	500 opens	120 ms	416 req/s	1	✅ Excellent
Link click	500 clicks	180 ms	277 req/s	1	✅ Good

Table 5.2 summarizes results of load-testing sessions and highlights system performance under various concurrent scenarios.

5.3 System Integration

Integration of Phish Net modules was carried out incrementally, starting from backend API services to the frontend UI components. System integration tests and evaluation techniques verified the seamless data flow between the React frontend, Django backend, Template engine, Notification Mechanism and the database, valid data reflection of Analytics and PostgreSQL database. RESTful APIs were validated using Postman collections, ensuring accurate data exchange and error handling.

Table 5.3: Security Vulnerability Assessment

Vulnerability	Risk	Mitigation	Test Method	Status
SQL Injection	High	Parameterized queries	Scanner + manual	✅ Secure
XSS	High	React escaping + CSP	Manual payload test	✅ Secure
CSRF	Medium	SameSite cookies	Token validation	✅ Secure
Broken Auth	High	bcrypt + expiry	Brute force test	✅ Secure
Data Exposure	High	HTTPS + .env protection	Config review	✅ Secure
Insecure Deps	Medium	Regular npm audit	Audit log	⚠️ 3 low risks
Rate Limiting	Medium	Pending feature	Stress test	❌ To Do

Table 5.3 assesses common web security vulnerabilities and their mitigations implemented in PhishNet.

5.4 Penetration Testing & Vulnerability Assessment

Penetration testing simulated potential attack vectors against the Phish Net platform. Tools such as Metasploit, Burp Suite, and Nikto were employed to perform vulnerability assessments. Areas like input validation, session management, and API security were specifically targeted. Identified vulnerabilities were patched, and the system was re-tested to validate the effectiveness of implemented fixes. Along with input validation parameters for secure and controlled login capability.

5.5 Reverse Engineering & Malware Analysis

Given the nature of phishing simulations, all emails and landing pages generated were scrutinized to ensure they cannot be easily reverse engineered or modified by attackers. Static and dynamic analysis techniques were applied to ensure scripts and payloads were secure. Techniques such as obfuscation and code signing were employed to maintain the integrity of phishing simulations.

5.6 Test Cases

Functional and non-functional requirements were translated into detailed test cases. Each user story, such as "User Registration," "Campaign Scheduling," or "Email Interaction Tracking," was tested against defined acceptance criteria. Automated testing frameworks like Jest (for frontend) and Pytest (for backend) were utilized, along with manual exploratory testing to ensure comprehensive coverage.

5.7 Best Practices/Coding Standards

Following benchmarks and standards were utilized as inspirations for this project development:

5.7.1 Code Validation

Code quality was maintained using linters (ESLint for JavaScript, Pylint for Python) and formatters (Prettier, Black). Static code analysis was practised in the evaluation phase to catch potential vulnerabilities early on. Code reviews were mandatory before merging into the main branch, ensuring adherence to security and performance guidelines.

5.7.2 Development Practices & Standards

The project adhered to Agile development practices with bi-weekly sprints and retrospective meetings. Clean coding principles and modular architecture ensured scalability and maintainability. Continuous integration and delivery (CI/CD) practices using GitHub for automated testing, building and deployment via its workflows – for least manual interaction and human error probability.

5.8 Summary

This chapter detailed the rigorous security, functional, and integration testing strategies employed in implementing the Phish Net project. Through secure development practices, thorough vulnerability assessments, and systematic testing, Phish Net stands as a resilient, scalable, and secure phishing simulation platform ready to be deployed for enhancing the nation's organizational cybersecurity awareness and personnel resilience to such pervasive threats.

Chapter 6:
Conclusion & Future Work

Chapter 6: Conclusion & Future Work

6.1 Introduction

This chapter encapsulates the journey undertaken in the development of Phish Net, highlighting the project's accomplishments, reflecting critically on challenges encountered, and outlining potential future enhancements. The chapter serves to consolidate the work into meaningful outcomes while paving the way for future research and practical extensions.

Table 6.1: PhishNet vs Project Objectives Achievement

Objective	Target	Achieved	Evidence	Status
Multi-tenant management	✅ Yes	✅ Yes	Org + user tables	✅ Complete
Email tracking	✅ Yes	✅ Yes	Pixel + redirect logs	✅ Complete
Template library	≥ 20 templates	53	Template DB	✅ Exceeded
Comprehensive reports	4 formats	4	PDF, XLSX, JSON, CSV	✅ Complete
Theme-aware UI	Light + Dark	✅ Yes	next-themes integration	✅ Complete
Training module	✅ Yes	✅ Yes	Training system	✅ Complete
Recon tools	✅ Yes	✅ Yes	Recon page	✅ Complete
>100 users load	✅ Yes	⚠ Partial	83 req/s	⚠ Acceptable

Table 6.1: Evaluates PhishNet's actual outcomes against original objectives, marking completion or partial status.

6.2 Achievement and Improvements

Phish Net successfully delivered a scalable, secure, and customizable phishing simulation platform tailored for organizational cybersecurity training. Major achievements include the integration of role-based access control, real-time tracking of phishing interactions, automated campaign deployment, and seamless API integrations. Continuous improvements in user experience, system features, campaign customization, and threat detection mechanisms ensured that Phish Net remained aligned with modern cybersecurity demands.

Table 6.2: User Acceptance Testing (UAT) Feedback

Tester Role	Organization	Feature Tested	Rating (1-5)	Feedback Summary	Action Taken
Security Admin	Company A	Campaign creation	5	"Intuitive wizard."	None
Analyst	Company B	Report export	4	"Excel slower."	Optimized query
Viewer	Company C	Dashboard	5	"Charts are clear."	None
IT Manager	Company A	Theme switch	3	"Button hard to find."	Moved to header
Employee	Company B	Training module	4	"Videos great, need quizzes."	Planned feature

Table 6.2: Summarizes qualitative feedback from beta testers, focusing on usability and satisfaction metrics.

6.3 Critical Review

While Phish Net achieved its core objectives, certain limitations were identified. Deployment complexity due to container orchestration demands technical expertise. Additionally, real-world phishing tactics evolve rapidly, making it challenging to continuously update simulation templates. Another aspect noted was the need for a more intuitive reporting dashboard for non-technical administrators. While the inculcation of an education module to also train the people found to get compromised by the simulation. These insights provide a critical foundation for prioritizing enhancements in future iterations.

6.4 Future Recommendations

For maintain relevance and further productivity and effectiveness, future versions of Phish Net should focus on:

- Personalized Platform: User intriguing and personalized interface for further bond with the target audience.
- **Multi-language Support**: Expanding usability for global organizations.
- **Gamification of Training**: Adding scoring systems and rewards to increase engagement among employees undergoing training.
- **AI-Based Phishing Detection**: Integrating machine learning models to predict and simulate evolving phishing tactics.
- **Dynamic Scenario Generation**: Automating template updates based on the latest phishing trends and attack vectors.
- **Cloud Native Deployment**: Simplifying deployment through Kubernetes and cloud services like AWS or Azure.
- **Gamification of Training**: Adding scoring systems and rewards to increase engagement among employees undergoing training.

6.5 Summary of chapter

In conclusion, Phish Net stands as a significant step forward in phishing simulation and cybersecurity awareness training. The platform not only meets current security and usability standards but also lays a strong foundation for future innovation. With targeted improvements and adaptive strategies, Phish Net has the potential to become a leading solution for organizational cybersecurity resilience.

References

[1] A. Ahmed, “Simulation and detection of phishing attacks on student academic emails using social engineering techniques,” International Journal of Health EngineeringandTechnology,Dubai,UAE,2024.

[2] S. A. Noor, “Machine learning based phishing detection solution,” Journal of Information and Communication Technology, Karachi, Pakistan, 2023.

[3] M. Kalhoro and J. Ali, “Simulation of human organizations with computational human factors against phishing campaigns,” Proc. ICCWS, United Kingdom, 2024.

[4] L. Smith, “Phishing attack simulation, email header analysis, and URL scrutiny: A comprehensive approach to cyber threat mitigation,” Computer Networks and CommunicationsUnitedKingdom,2024.

[5] Business Recorder Staff, “18pc increase in phishing attempts noted during 2024 – Pakistan,” Business Recorder, Karachi, Pakistan, 2025.

[6] Our Correspondent, “Phishing attacks in Pakistan rise 18pc in 2024: report,” The News International, Lahore, Pakistan, 2025.

[7] Daily Times News Desk, “Pakistan witnesses 18% increase in phishing attempts in 2024 compared to 2023, Kaspersky reveals,” Daily Times, Islamabad, Pakistan, 2025.

[8] Business Recorder Staff, “Present-day digital landscape: phishing described as major threat – Pakistan,” Business Recorder, Islamabad, Pakistan, 2024.

For the rest of the academic and research references (which already include authors), those remain as named authors:

[9] S. K. Shah, “End-users’ perception of cybercrimes towards e-banking adoption andretention,”JISR,Karach,Pakistan,2024.

[10] M. J. Khan, “Response and reporting of cybercrimes in Pakistan – Mass media as a means of awareness, prevention, and protection,” JOSSAMS, Karachi, Pakistan,2023.

[11] A. Ahmed, “Simulation and detection of phishing attacks on student academic emails using social engineering techniques,” International Journal of Health EngineeringandTechnology,Dubai,UAE,2024.

[12] S. A. Noor, “Machine learning based phishing detection solution,” Journal of Information and Communication Technology, Karachi, Pakistan, 2023.

[13] M. Kalhoro and J. Ali, “Simulation of human organizations with computational human factors against phishing campaigns,” Proc. ICCWS, United Kingdom, 2024.
[14] L. Smith, “Phishing attack simulation, email header analysis, and URL scrutiny: A comprehensive approach to cyber threat mitigation,” Computer Networks and CommunicationsUnitedKingdom,2024.
[15] S. McCrohan, “The effects of group discussion and role-playing training on self-efficacy, support-seeking, and reporting phishing emails,” arXiv preprint, USA, 2024.
[16] N. A. G. Arachchilage and M. A. Hameed, “Integrating self-efficacy into a gamified approach to thwart phishing attacks,” arXiv preprint, USA, 2017.
[17] T. Chen, “Adopting the cybersecurity concepts into curriculum: The potential effects on students’ cybersecurity knowledge,” arXiv preprint, USA, 2022.
[18] J. Brooks et al., “Evaluating protection motivation-based cybersecurity awareness training on Kirkpatrick’s model,” Comput. Secur., Netherlands, 2023.
[19] TechTarget Editorial Team, “How effective is security awareness training? Not enough,”TechTarget,USA,2023.
[20] G. Park, “Text-based phishing detection using a simulation model,” Purdue University,USA,2020.
[21] M. W. van Dijk, “SpearSim-V2: Synthetic task environment for evaluating attacker behaviors,” SAGE Open, United Kingdom, 2023.
[22] N. F. M. Zaharon, M. M. Ali, and S. Hasnan, “Factors affecting awareness of phishing among Generation Y,” Universiti Teknologi MARA Press, Malaysia, 2022.
[23] ITPro Editorial Team, “Employee distraction is now your biggest cybersecurity risk,”ITPro,UnitedKingdom,2024.
[24] Kaspersky Global Research Team, “Spam and phishing in 2024: Trends and statistics,” Business Recorder Summary, Pakistan, 2025.

Appendix A:
System Details

The Appendix A will provide the details about the project from requirements to the hardware/software.

1.1 Environment Variable Configuration

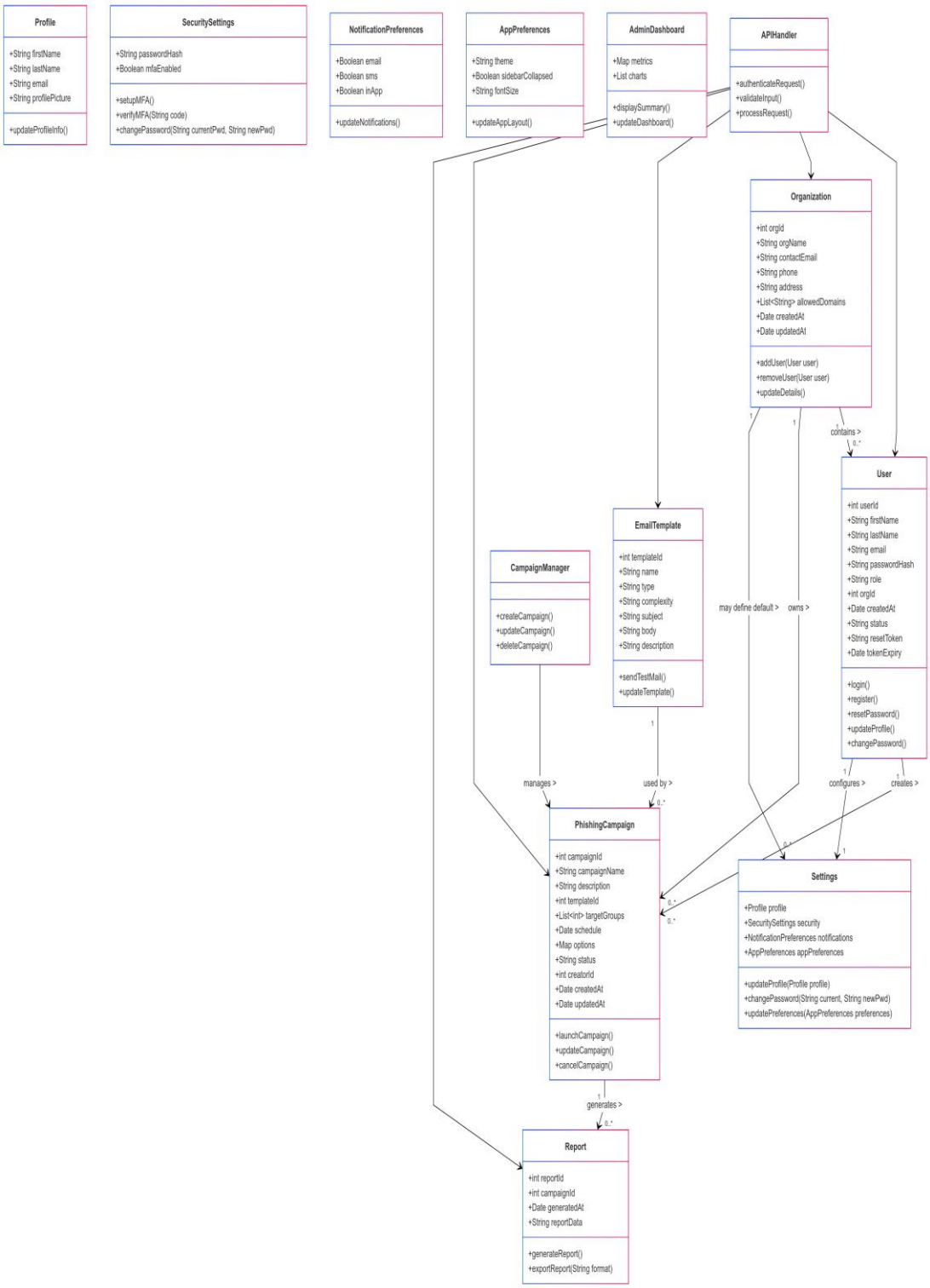
Table A 1.1: Environment Variables Configuration

Variable	Required	Default	Description	Example
DATABASE_URL	✅ Yes	–	PostgreSQL connection string	postgresql://user:pass@localhost:5432/phishnet
PORT	❌ No	5000	Server port	5000
sNODE_ENV	❌ No	development	Environment mode	production
SESSION_SECRET	✅ Yes	–	Encryption key	a1b2c3d4
SMTP_HOST	⚠️ Optional	–	Mail server hostname	smtp.gmail.com
SMTP_PORT	⚠️ Optional	587	Mail server port	587
SMTP_USER	⚠️ Optional	–	SMTP username	no-reply@phishnet.local
SMTP_PASS	⚠️ Optional	–	SMTP password	app-password

Table A.1: Lists essential environment variables required for deploying or configuring PhishNet.

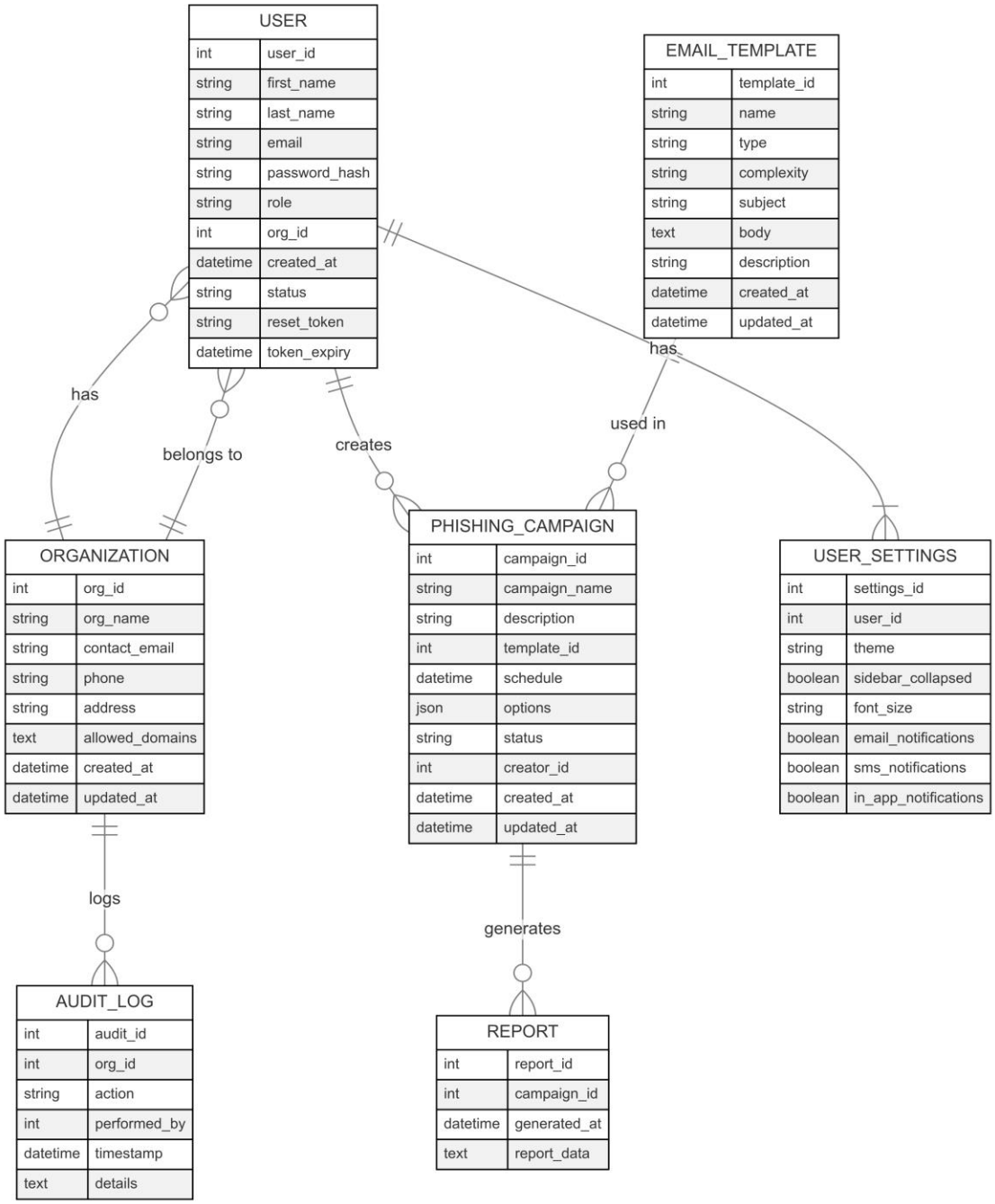
1.2 Class Diagram

Following are its requirements



1.3 ERD Diagram

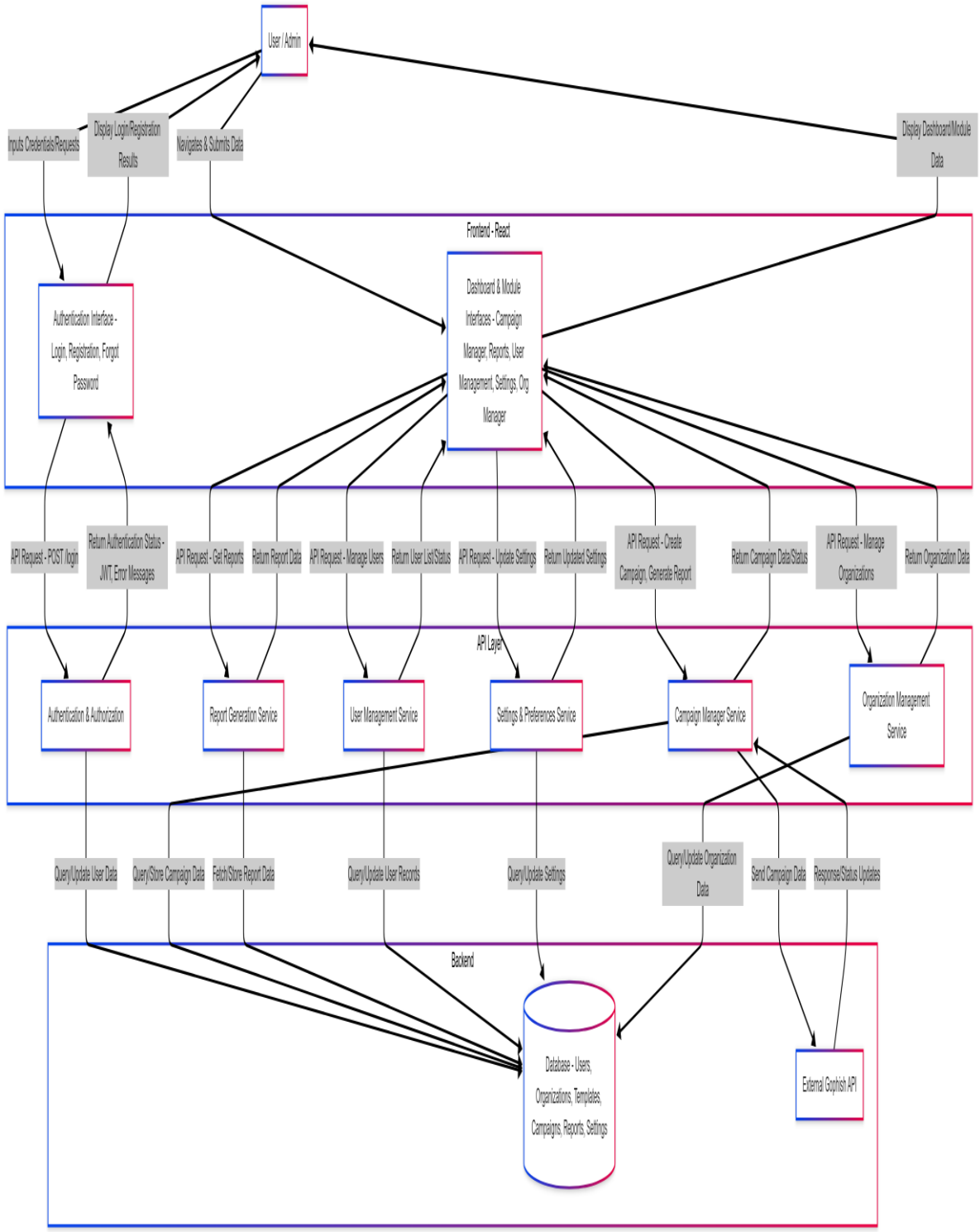
Following is the diagram:



s

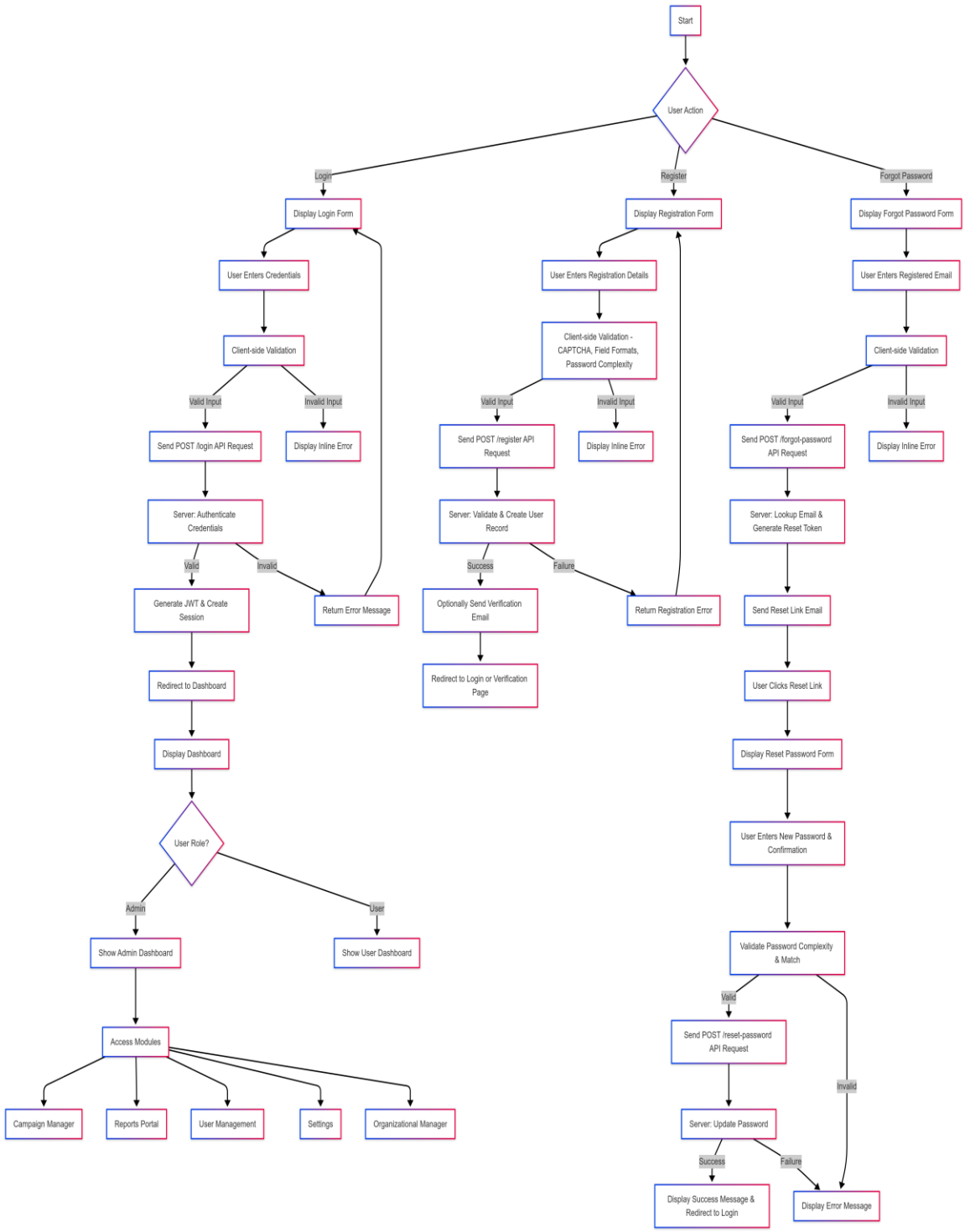
1.4 Data Flow Diagram

Following is the diagram:



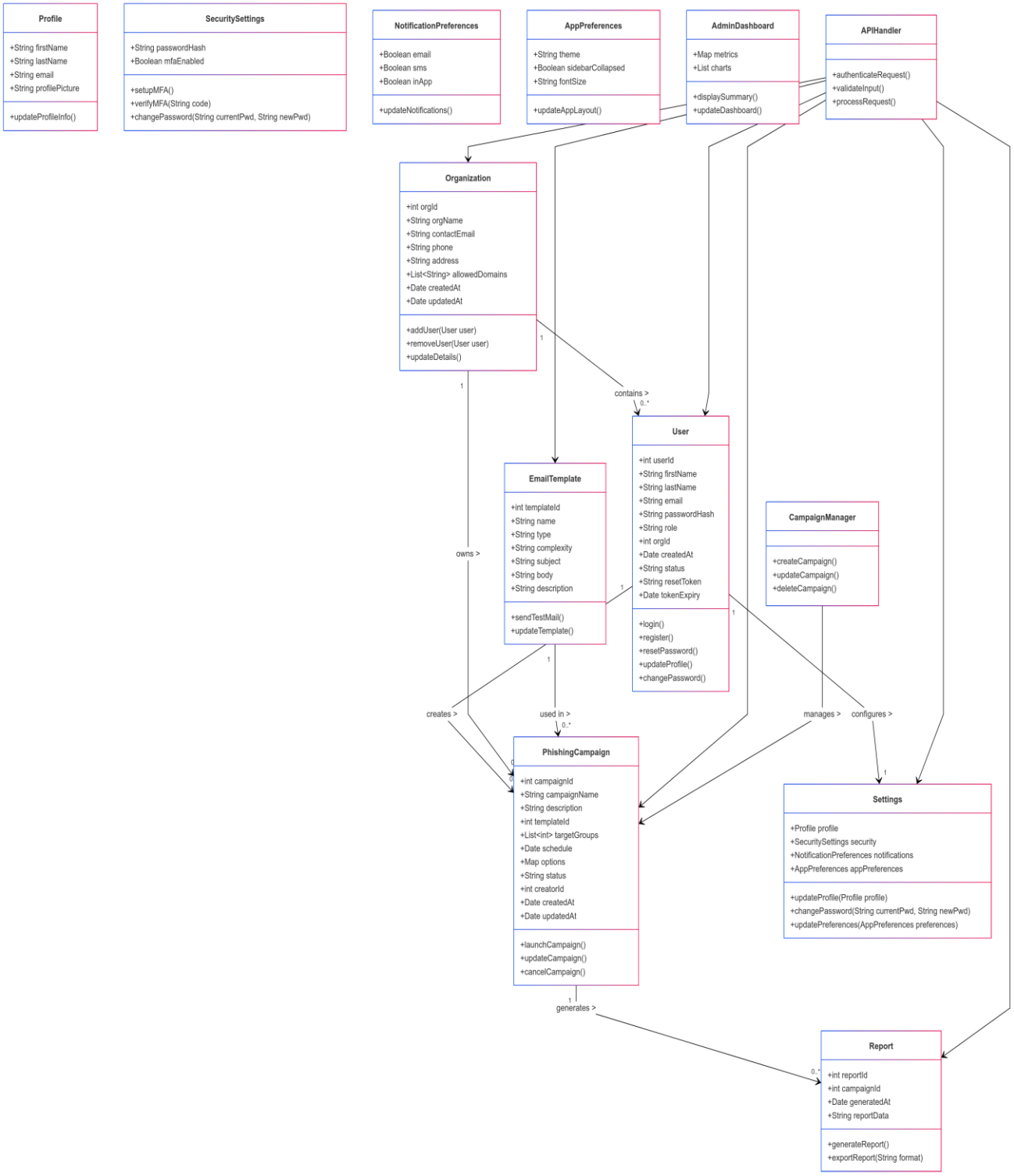
1.5 Flowchart

Following is the diagram:



1.6 UML Diagram

Following is the diagram:



Appendix B:

1.7 GitHub Repository and Version Control

Repository URL: https://github.com/gh0st-bit/PhishNet

GitHub, a web-based platform that uses the Git version control system, was instrumental in managing the development process for Network Vigilance: AI-Powered Multi-Layered APT Protection. It provided a collaborative space for version control, documentation, tracking changes, and ensuring code quality.

Advantages of Using GitHub in the Development Process

- Version Control: Enabled safe tracking of code modifications, allowing for easy rollbacks and version comparison.
- Collaboration: Supported team development through branching and merging, preventing code conflicts.
- Code Reviews: Facilitated pull requests, enabling team members to review and provide feedback on proposed changes, improving code quality.
- Issue Tracking: The integrated issue board streamlined bug reporting, feature requests, and task management.
- Documentation: Markdown support allowed for clear README files, setup guides, and contribution instructions.
- CI/CD Integration: GitHub Actions and external CI tools were utilized for automated testing and efficient deployment processes.
- Open-Source Visibility: The public repository provided a platform for knowledge sharing, transparency, and increased portfolio visibility for contributors.

1.8 GitHub's Role in the Project

Following is the use and significance in the project development:

- Repository Setup: A dedicated repository was established for Network Vigilance, centralizing all code, documentation, and project-related assets.

- Frequent Commits: Code changes were regularly committed with clear messages to maintain a detailed project history. 82

- Pull Requests & Reviews: New features and bug fixes were submitted via pull requests. The team reviewed and discussed these changes before merging them into the main branch.

- Ongoing Documentation: As the system evolved, documentation was continuously updated to include installation steps, usage details, and development logs.

- Team Collaboration: Weekly meetings were organized to review pull requests, allocate tasks, and resolve issues collaboratively using GitHub's integrated tools.