



University of Arkansas – CSCE Department Capstone I – Final Proposal – Fall 2025

PhishWise Scam Trainer

Paul Tribble, Milo Pumford, Sarah Smith, Shane Norden, Victor Berrios, Warren Olvey

Abstract

Phishing scams are a widespread and persistent cybersecurity threat, with an estimated 3.4 billion phishing emails sent every day and major providers like Google blocking around 100 million of them daily [9]. Despite these protections, many individuals still fall victim to phishing attempts. Existing training solutions, such as KnowBe4 or Sophos, are designed for corporations that can afford large-scale services, leaving the average user without accessible tools for cybersecurity awareness.

PhishWise is a web-based phishing awareness and training system designed to help individuals and households recognize and avoid online scams through realistic, personalized learning modules. The platform sends simulated phishing emails based on common scam patterns and redirects users to short educational modules when they interact with them. By focusing on individuals, families, and small groups rather than large organizations, PhishWise provides a more personal and accessible training experience. The system will be hosted on a Vercel server for development and testing, allowing scalability for future design improvements and public deployment. By offering an affordable and tailored training environment, PhishWise aims to help users become more aware of phishing attempts and protect their sensitive information from potential attackers.

1.0 Problem

Phishing has been described as the “most pervasive way of implementing social engineering” to obtain personal and sensitive information [13]. It remains a significant threat across all demographics. An estimated 3.4 billion phishing emails are sent daily, and Google blocks approximately 100 million phishing messages each day [9]. However, many users still encounter harmful attacks. A survey by the Pew Research Center reported that 73% of U.S. adults have experienced credit card fraud, ransomware, or online shopping scams, with 32% reporting such incidents within the past year [10]. Globally, phishing accounted for 83% of all cyber-attacks on businesses in the United Kingdom in 2022, and it was the most common cyber-attack type

targeting Asian organizations in 2021 [9]. Financial losses are also rising, with phishing-related damages increasing by 74%—reaching \$1.7 billion—from 2020 to 2021 [11].

Older adults are particularly vulnerable. The FBI’s Internet Crime Complaint Center (IC3) reported that individuals aged 60 and older filed nearly 150,000 cybercrime complaints in 2024, with losses approaching \$5 billion, far exceeding any other age group [5]. Research shows that while older adults are aware of cybersecurity risks, they have greater difficulty distinguishing legitimate emails from phishing attempts [1]. These findings highlight the need for accessible tools that can help users—especially older individuals—identify phishing attempts and protect themselves.

Large corporations often use services like KnowBe4 and Proofpoint to provide phishing awareness training for employees. However, these platforms are expensive and are not designed for individuals or small groups. KnowBe4, for example, offers plans starting at \$3.25 per person for groups of 25–50 users but does not provide affordable options for independent users or families [14]. This pricing structure limits personalized training and leaves everyday users without practical resources for improving their cybersecurity awareness.

As phishing attacks grow in frequency, realism, and impact, the lack of accessible training options creates a significant problem: How can the average user become more aware of cyber threats and avoid falling victim to phishing attacks in an affordable, practical way?

2.0 Objective

The objective of PhishWise is to develop a functional prototype of a web-based phishing training system that simulates real-world phishing attacks through email. The user will be able to create an account and either create or join a group called a “school”. The manager of the school will determine the frequency of the phishing attempts, then review the user data in the network. When the user clicks on a link sent by the system, they will be redirected to a training module to help the user become educated and more aware of their failure. PhishWise will store data to showcase the progress of the user to the manager. We will host the system on a Vercel server for testing and internal deployment as well as design a scalable backend for later future public release.

3.0 Background

3.1 Key Concepts

Email

Email is a common method for exchanging messages over the internet using a client–server architecture. Email clients such as Gmail or Outlook communicate with mail servers using standard protocols like SMTP (Simple Mail Transfer Protocol), IMAP (Internet Message Access Protocol), or POP3 (Post Office Protocol) to send and retrieve messages. Each email includes headers, which contain information such as the sender, recipient, and subject, as well as a body containing the message content. Phishing attacks often manipulate headers to impersonate legitimate individuals or organizations and may embed malicious links or misleading content within the email body. Because email is widely used across personal, business, and government

contexts, it is a frequent target for attackers and a common delivery mechanism for phishing scams.

Phishing

According to the National Institute of Standards and Technology (NIST), phishing is “a technique for attempting to acquire sensitive data, such as bank account numbers, through a fraudulent solicitation in email or on a web site, in which the perpetrator masquerades as a legitimate business or reputable person” [4]. Phishing attacks often combine several tactics to increase their effectiveness. One common tactic is bait creation, where attackers craft messages that closely resemble communications from trusted organizations to convince victims the message is legitimate [3]. Another common tactic is social engineering, which relies on psychological manipulation to influence victims into revealing sensitive information. Attackers may gather publicly available information to appear credible, impersonate trusted contacts, and gradually lower a victim’s skepticism until the victim completes the action the attacker intends.

Social Engineering

Social engineering is a manipulation technique that exploits human psychology and human errors to obtain sensitive information. An attacker typically begins by gathering publicly available information about a victim to craft a convincing pretext. Once sufficient information is collected, the attacker initiates contact and establishes credibility by impersonating a trusted individual or organization. Subsequent interactions rely on psychological manipulation to lower the victim’s guard until they eventually perform the action the attacker intends. The attack succeeds at the point when the victim complies, such as by clicking on a malicious link, sharing personal information, or granting unauthorized access.

3.2 Related Work

Many organizations use cybersecurity training programs that include simulated phishing attacks, but these solutions are typically designed for large enterprises. They are often internally developed or bundled within expensive security packages, making them inaccessible to individuals, retirees, families, or small groups. One of the core goals of PhishWise is to provide similar training capabilities in a more accessible and affordable form.

Microsoft Defender

Microsoft Defender is widely used and includes several tools to protect users from phishing attacks. In its paid Plan 2 subscription, organizations can administer simulated phishing campaigns to employees [7]. These simulations provide detailed metrics and feedback for administrators. However, this feature is part of a larger enterprise security suite and is not intended for individuals or small-scale users, limiting its accessibility outside organizational environments.

PhishingBox

PhishingBox is another paid platform used by companies to train employees to identify phishing emails. It offers a small free test that allows users to check how well they can spot phishing attempts. However, the free test has limitations: it consists of only ten questions, reshuffles the same examples on repeat attempts, and provides minimal explanation about why each message is legitimate or malicious. Additionally, it operates in a controlled quiz environment rather than

delivering real emails to a user's inbox. PhishWise aims to address these limitations by delivering varied phishing simulations, offering immediate explanations after failures, and sending messages at unpredictable times to better reflect real-world conditions.

CanIPhish

CanIPhish provides simulated phishing attacks for organizations and includes both free and paid tiers [6]. It offers micro-trainings for individuals who fall for simulated attacks and provides administrators with detailed feedback. While CanIPhish includes many elements relevant to PhishWise, it still requires an administrator to configure and issue campaigns to users. PhishWise will support administrators as an option, but it will not rely on them. The platform is designed so individuals or small groups can receive automated phishing simulations without requiring manual campaign setup, making the system more accessible for everyday users.

4.0 Design

4.1 Use Cases

The PhishWise platform supports three core user roles: general users, managers, and administrators. Each role interacts with the system differently, contributing to a layered structure for simulation delivery, training, oversight, and system maintenance. The platform is designed to provide personalized phishing awareness training while also supporting small-group management through "schools."

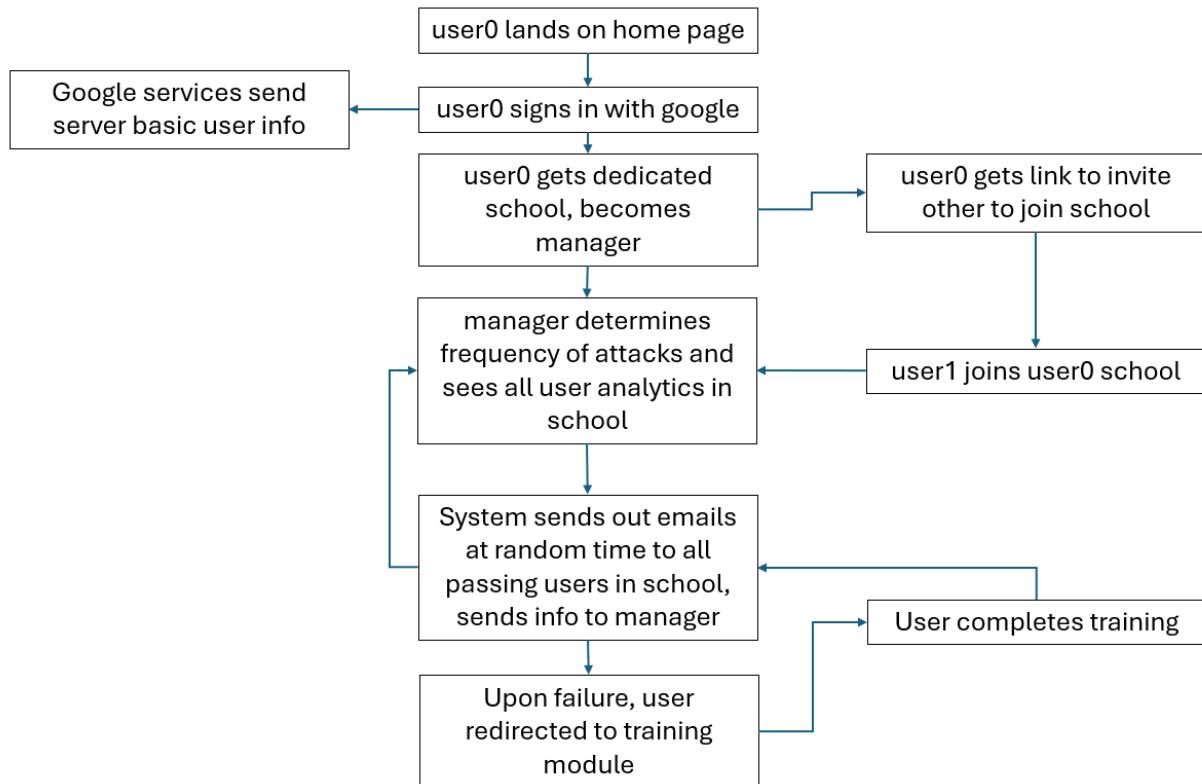


Figure 1

General Users

General users are the primary recipients of phishing simulations. These users register through the PhishWise web application using Google OAuth 2.0 for account creation and authentication. After registering, each user begins receiving randomized phishing simulations delivered through email, based on predefined template sets. Additional simulations may be sent automatically if a user repeatedly fails certain phishing types, or manually if a manager assigns supplemental training.

When a user clicks a simulated phishing link, the system logs the interaction and redirects the user to a targeted training module associated with the template they failed. All user actions, such as email opens, link clicks, and training completions, are stored in the PostgreSQL backend and tied to their unique Google ID for long-term tracking. Users have access to a personal dashboard where they can review their simulation history and performance metrics.

Users may participate independently or choose to join a group structure called a “school.” A user may only belong to one school at a time under a given account, but all account progress persists even if they leave and later rejoin.

Managers

Managers begin as general users but obtain elevated permissions once they create a school or are assigned a managerial role within one. Managers oversee the performance of all users in their school and have access to an expanded dashboard containing analytics for every member. From this dashboard, they can monitor simulation outcomes, track user improvement, assign targeted training modules, and adjust the frequency of phishing simulations delivered to their group. Managers may also distribute optional supplemental simulations to address specific weaknesses or training needs.

Additionally, managers are responsible for adding new members to their school. They accomplish this by sending secure, email-based join codes that allow users to link their accounts to the school. Managers do not receive phishing simulations by default, but they may opt into receiving them if they want to observe and experience the same training pipeline as the users they oversee.

Administrators

Administrators are responsible for maintaining the full PhishWise platform at the system level. They do not participate in phishing simulations and are not responsible for managing individual users or schools. Instead, their role focuses on technical and operational maintenance of the platform. Administrators manage the email-sending infrastructure, maintain the phishing template library, update training module content, and oversee the backend scheduling mechanisms that determine when simulations are delivered. They also ensure the stability of the system’s architecture, maintain integrations such as SendGrid and LMS APIs, and handle security mechanisms such as token-based link tracking and redirect logic.

Administrators also have access to anonymized platform-wide performance data. This allows them to evaluate usage patterns, identify trends, and make adjustments that improve system

efficiency or training effectiveness. Their work ensures that the platform operates reliably, scales effectively, and remains aligned with current phishing and cybersecurity practices.

PhishWise is designed to function with minimal manual involvement once initial setup is complete. Simulations are scheduled automatically and adapt to each user's performance patterns. The interaction between the frontend, token tracking, redirect logic, and training completion records forms a closed learning loop that supports continuous user improvement. Persistent storage in the backend database ensures that progress records remain intact regardless of school membership changes, allowing for long-term behavioral tracking across the platform.

User Roles and Capabilities

Role	Receives Simulations	Can View Metrics	Can Assign Training	Can Create Schools	Can Invite Users	Has System Access
General User	Yes (by default)	Own data only	No	Yes (becomes mgr)	No	No
Manager	Optional	All users in school	Yes	Yes	Yes	No
Admin	No	All schools	Yes (global content)	No	No	Yes

4.2 Requirements

4.2.1 Functional Requirements

Each PhishWise instance will collect and process several categories of user and system data to simulate realistic phishing conditions, track user responses, and deliver adaptive training content. The system handles three main data types: identification, behavioral interactions, and performance analytics.

Identification data includes the user's email address and Google OAuth 2.0 credentials, which are stored securely upon registration. This information serves as the target endpoint for email simulations and as the primary key for linking records in the PostgreSQL database.

Behavioral interactions represent how users respond to simulated phishing attempts, such as opening an email or clicking a link. These interactions are captured in real time through embedded tracking tokens and redirect URLs. When a user clicks a simulated phishing link, the backend immediately logs the event and redirects the user to a training module designed for the specific template they interacted with.

Performance analytics are derived from accumulated behavioral interactions. These data points are stored in the backend and used to evaluate long-term user awareness and improvement.

Repeated failures of the same template type may trigger additional modules or increased simulation frequency, while consistent correct identification may gradually increase difficulty.

Each phishing event follows a standardized flow. The system selects a template from the predefined library, formats the message, and dispatches it via SendGrid. Engagement metrics such as open rate and click rate are recorded as users interact with the email. Training modules are automatically delivered when a user fails a simulation, providing an explanation of missed red flags and guidance on identifying similar attacks.

All training and simulation data will be timestamped, versioned, and logged in a centralized PostgreSQL database hosted within the Vercel environment. This design supports long-term trend analysis, detection of repeated mistakes, and school-level analytics for managers. A user dashboard will display simulation history, performance graphs, and completion status of assigned modules. Managers will have access to a similar dashboard for their school, where they can adjust simulation frequency, monitor member progress, and assign supplemental training modules as needed.

System inputs include template selections, user identification data, and interaction logs. Outputs include email deliveries, module redirects, metrics dashboards, and performance summaries. All data will be validated and encrypted, with anonymization applied to administrator-level analytics as required for privacy and compliance.

4.2.2 Interface Requirements

The PhishWise platform acts to create a seamless, secure, and automated experience for cybersecurity awareness and training. Using third-party communication APIs like Twilio API, GraphQL, and SendGrid, the system will have a modern responsive web interface and robust backend services.

To ensure the user experience is simple and accessible, the fully responsive web interface will remain standardized across desktops, tablets, and mobile devices. With dynamic web development, CSS and HTML allow screen size adjustment. Doing this will help to make sure the web interface has the correct proportions for each device. Keeping that in mind, this will be tested extensively throughout development with Chrome Dev Tools. This will set the screen to mimic each of the different screen sizes. The interface itself will be created in TypeScript with the React framework to ensure it is resilient, scalable, and bug resistant. Considering speed and efficiency are prioritized, GraphQL will be used to communicate between the front-end and back-end. When the front-end sends a request for data, GraphQL will receive the request, the data will then be fetched, and the query will be executed. Using GraphQL will allow us to obtain the exact data we need without any extra and minimal mapping.

For emailing, SendGrid will be used to send out the emails at the user's request at sign up. This cloud-based email service allows for sending, receiving, and tracking without running our own email server. To connect SendGrid to the web interface, an API key links the two. The front-end will handle the sign-up form, and the backend will call the API. The API has a two-factor authentication choice which will send messages to authorize the user. This will all be connected through GraphQL endpoints.

PhishWise aims to spread information on cybersecurity. If a user is to respond to the email, this will direct them into the training content we have provided. To manage our educational content, TalentLMS will relate to an API. TalentLMS is a cloud-based learning platform to help organize and deliver learning content to your users. This will hold our personalized content as well as provide feedback on the performance of our users.

PhishWise will be using Google Cloud's OAuth 2.0 to ensure a smooth sign in experience for users. This approach minimizes sensitive data that must be stored on PhishWise servers, as well as making it much simpler for users to sign in.

4.2.3 Performance Requirements

PhishWise should use minimal resources in order to serve the most traffic as efficiently as possible. Email generation should be timely and should be sent out at random times each week to ensure users are constantly monitoring their inboxes for potential attacks. When a user engages with a simulated attack, our server should quickly reroute them from the PhishWise attack page to the appropriate education module. To reduce load on the PhishWise servers, these modules should be rendered on the client-side as much as possible.

4.3 High Level Architecture

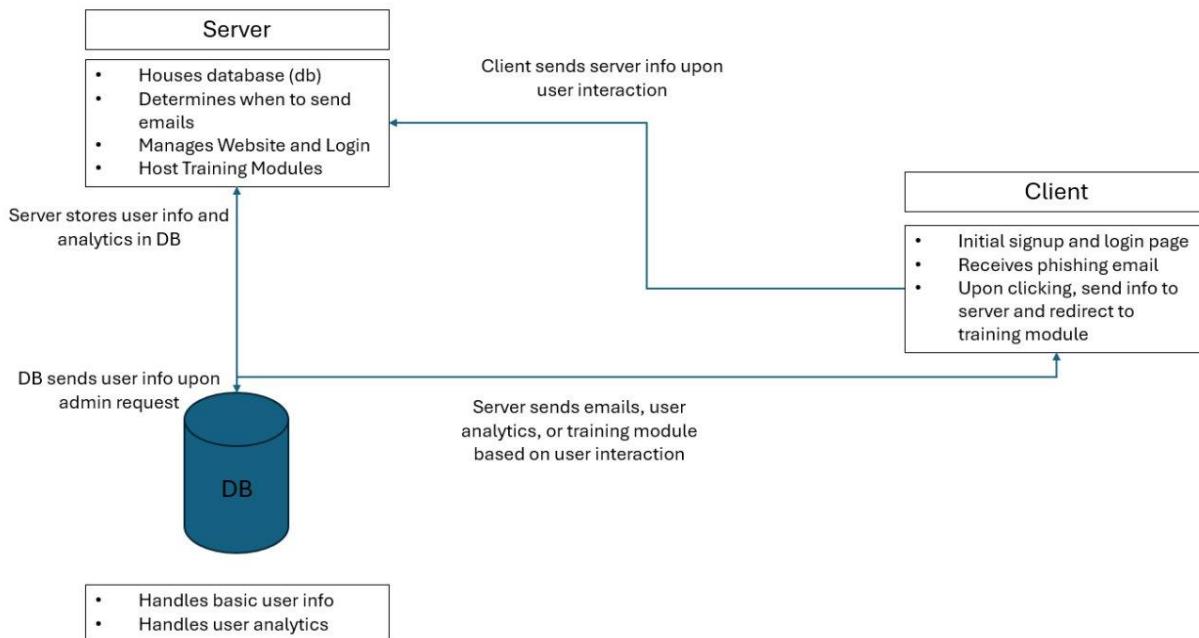


Figure 2

PhishWise follows a two-tier client-server architecture, with a web client running in the user's browser and a server tier hosted on Vercel that includes the application logic and a connected PostgreSQL database. At a high level, the client is responsible for rendering the user interface, handling sign-up and login via Google OAuth 2.0, and displaying dashboards and training modules. The server is responsible for managing user data, scheduling and sending phishing simulations, logging interactions, and serving training content.

As shown in Figure 2 (system overview diagram), the client initiates interactions by sending requests to the server whenever a user signs up, logs in, or clicks a link in a phishing email. During signup or login, the client redirects the user to Google for OAuth authentication. Once authenticated, Google returns an authorization code to the PhishWise server, which exchanges it for a token and extracts the user's Google ID. The server then stores this identifier and any associated user metadata in the PostgreSQL database.

For phishing simulations, the server uses scheduled logic to select an email template and dispatches the email to the user through SendGrid. When the user clicks a simulated phishing link, the request is routed back to the server, which logs the event, looks up the associated user and template, and then redirects the client to the corresponding training module. Managers access an extended version of the web client that queries the same server and database for aggregated analytics for their school. In all cases, the database acts as the central store for user accounts, simulation history, and training progress, while the server mediates all communication between the client and the data layer.

4.4 Detailed Architecture

The PhishWise web application handles onboarding of new users, delivery of phishing simulations, and management of user analytics. The front-end layer is built in TypeScript using the React framework, providing a responsive and accessible UI for both general users and managers. This layer renders the landing page, login and sign-up views, training modules, and manager dashboards. Example screens are shown in Figure 3 and Figure 4, which illustrate how users navigate between registration, training, and analytics views.

Between the front end and the data layer is the application layer, which runs on the Vercel-hosted server. This layer exposes a GraphQL API that handles all core operations, such as retrieving user metrics, updating training progress, and managing school memberships. It also integrates with SendGrid for sending registration emails and phishing simulations, and with Google OAuth 2.0 for authentication. Internally, the backend is organized into modules that manage user accounts, simulation scheduling, template selection, and event logging.

User and simulation data are stored in a PostgreSQL database connected to the server. The database contains tables for users, schools, email templates, simulation events, and training module completions. The application layer uses an ORM or query layer to read and update these tables in response to API calls from the client. For example, when a user clicks a simulated phishing link, the backend records the event in the simulation events table, looks up the associated template, and then returns the correct training module content to the client.

Authentication is handled using Google Cloud's OAuth 2.0. The developers configure the OAuth consent screen and credentials in the Google Cloud Console. When a user chooses to sign in, the client redirects them to Google's login page. After successful authentication, Google returns an authorization code to the PhishWise server, which exchanges it for an access token. The server reads the user's Google ID from this token and uses it as the primary identifier in the database. PhishWise stores only the Google ID and the metrics required for training and analytics, relying on Google to manage passwords and other sensitive credentials. This approach reduces the

security burden on the application while still allowing persistent user accounts and progress tracking.

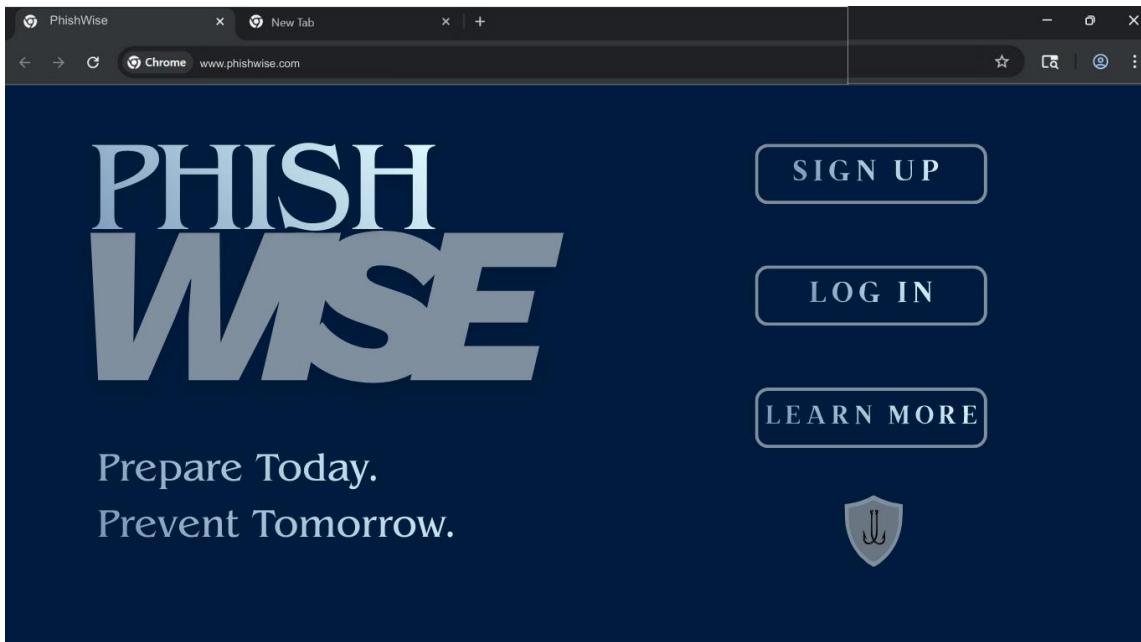


Figure 3

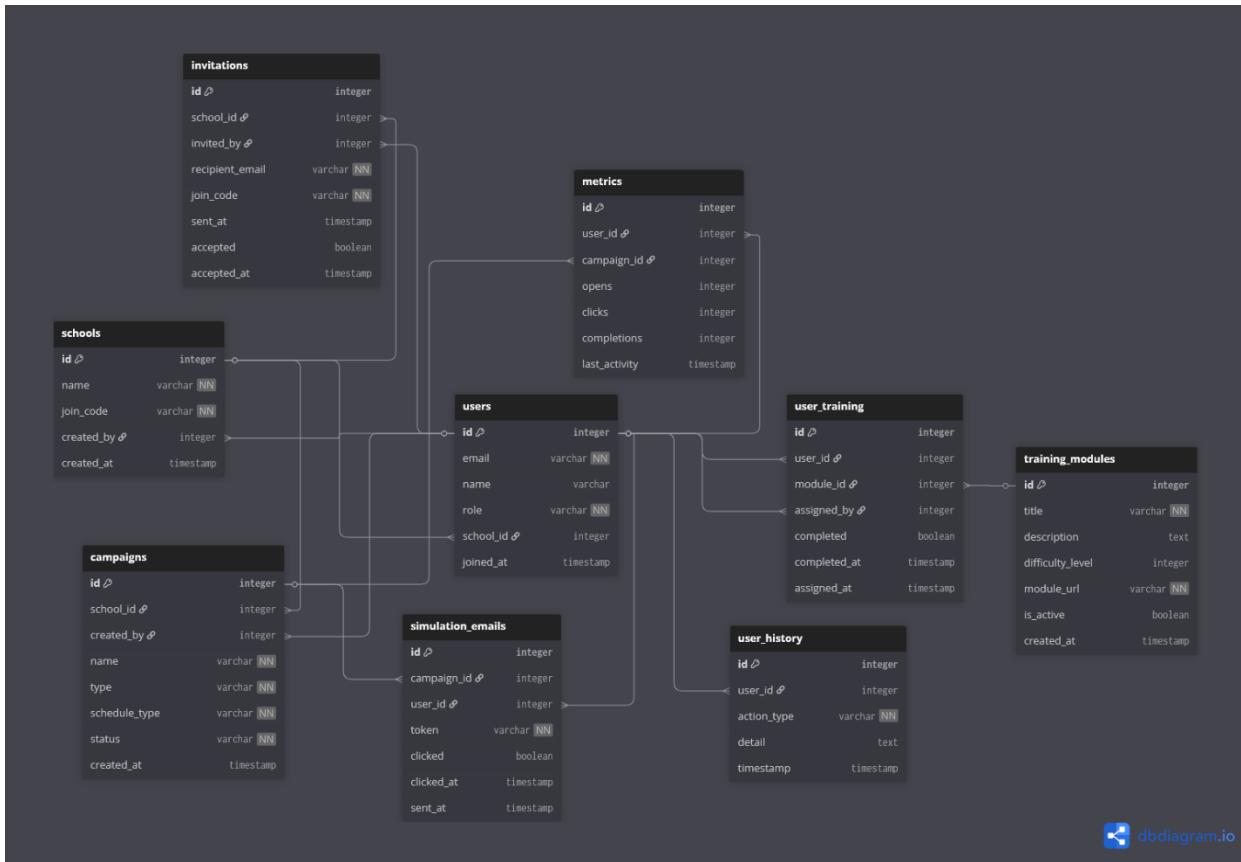
A screenshot of the PhishWise dashboard. The left sidebar contains navigation links: "HOME", "SETTINGS", "MANAGE USERS", "USERS" (with four entries: "Username 1" through "Username 4"). The main content area has a sidebar with a user profile picture and the greeting "Hello [Insert Name Here]". The main content is divided into several sections: "ANALYTICS" (User Progress chart showing a downward trend), "SUMMARY" (text placeholder: "INSERT DATA SUMMARY BELOW. IDEALLY HIGHLIGHTING THE OVERALL PERFORMANCE OF THE GROUP"), "NEXT HIT" (Date: 11/23/2025, Method: Email), and "USER SUMMARY" (three entries: "Username 1: Clicks: 2 - Lesson: Complete", "Username 2: Clicks: 0 - Lesson: Complete", "Username 3: Clicks: 5 - Lesson: Incomplete"). The browser's address bar shows "www.phishwise.com".

Figure 4

4.5 Data Management and Storage

Our database will be implemented using PostgreSQL and stored on our server. This database will store basic information about the users and will identify which users are managed by other users.

The database will also contain a record of emails sent, the status on if the user that email was sent to has fallen for the phishing attack, as well as what learning module the email belongs to. This will allow the website to calculate the analytics for any given user and direct that information to the user themselves or to the manager that the user is under.



[Click here to view database schema](#)

The templates for the emails will be categorized into learning modules that each focus on a different type of phishing attack or tactic used in phishing. These templates will be stored in the server, along with the material for the associated training module that the user will be linked to whenever they fail an email of that template type.

4.6 PhishWise Curriculum and Templates

The PhishWise training curriculum is designed to teach users how to recognize and avoid phishing attacks using a concise, structured approach based on credible, industry-standard guidance from the Cybersecurity & Infrastructure Security Agency (CISA) and the National Institute of Standards and Technology (NIST). These sources include generally accepted best practices for identifying phishing emails, understanding common social engineering tactics, and maintaining digitally safe habits.

4.6.1 Training Modules

Each module will address a different kind of phishing attack, based on definitions by CISA and NIST. The modules will be short and focused, so users can learn efficiently without requiring

large amounts of time. Each PhishWise module structure will follow a common seven-section format for clarity, accessibility, and consistency in learning. The planned module structure is:

1. Overview: A short description of the phishing scam type, why attackers use them, and how common they are.
2. Social Engineering Tactics: Teaches the psychological manipulation techniques common in this scam category.
3. Red Flags: Lists the key warning signs that users should watch for in similar emails, according to CISA and NIST standards.
4. Attack Objective: Describes what attackers usually want in this phishing type.
5. Examples: Gives common examples of this type of scam and points out the red flags to look for as discussed above.
6. Prevention Steps: Outlines simple, actionable prevention steps for promoting safer email behavior.
7. Practice Check: Short multiple-choice quiz reinforcing the concepts from the module. If the user came from failing a phishing test, then the practice check will be required.

4.6.2 Instant Feedback

PhishWise uses an instant feedback system to provide feedback when a user fails a simulated phishing test. When a user clicks a phishing link, they are sent to a short page that informs them they have failed the phishing test, highlights the key indicators they missed, and explains the social engineering concepts in the message. This helps users connect their action to the underlying phishing tactic, improving recognition and retention. From there, the user is guided directly into the relevant training module after reviewing the feedback, providing a seamless transition from the simulation to the learning experience.

4.6.3 Templates

PhishWise uses a library of phishing email templates to generate the simulations users receive. These templates pull in curriculum concepts from the modules and mimic the types of phishing patterns outlined by sources such as CISA and NIST. Each template has a flexible structure, with some parts being fill-in-the-blank and others offering multiple options, so we can create a wide range of emails and make sure users are not seeing the same thing every time. This variety allows for realistic emails and gives users another way to apply what they learned in the curriculum through real-world practice.

Each training module will have a set of templates that managers can choose from. This relationship links the emails sent to the learning modules. When a user fails a phishing attempt, the link tells the system what training module to assign to help them review the concepts they missed. This ensures there is a clear connection between the simulations the users encounter and the material they learn afterward.

Each role interacts with the template library differently. Managers can browse available templates when choosing which phishing simulations to assign or when they want to focus on specific scam patterns within their group. Administrators maintain and update the template library by adding new templates, updating existing templates, and ensuring the examples remain aligned with current phishing trends. This allows PhishWise to be flexible and stay relevant as phishing scams evolve.

5.0 Development Plan

5.1 Tasks

Content Team (Templates and Modules)

- Finalize list of phishing categories (such as credential theft, invoice scams, shipping scams).
- Draft learning outcomes for each training module.
- Write first draft lesson plan text for each module.
- Create example “failed email” screenshots or mockups.
- Write red-flag explanations for each template category.
- Write the full text for each training module (examples, descriptions, tips).
- Write phishing email templates for each category.
- Insert tracking placeholders into templates for backend integration.
- Tag templates with the module IDs they correspond to.
- Upload templates and training content into the server.
- Revise templates based on backend feedback during scheduling tests.
- Create help text for training modules and dashboards.
- Review all module content for consistency and clarity.

Backend Team (Server, Database, APIs, Emails)

- Set up the Vercel server environment and initialize the project.
- Configure environment variables for PostgreSQL, SendGrid, and OAuth.
- Implement the PostgreSQL schema (users, schools, templates, modules, events).
- Create initial seed data for templates and modules.
- Implement Google OAuth login and callback endpoints.
- Store and validate the Google ID for each user.
- Set up the GraphQL server with core queries and mutations.
- Add resolvers for user data, school membership, and template lookups.
- Add mutations for logging events and updating training progress.
- Integrate SendGrid for sending emails.
- Add tokenized tracking links to simulation emails.
- Build redirect endpoint to log clicks and retrieve module IDs.
- Implement simulation scheduling (intervals, randomization, manager settings).
- Implement school creation logic and join-code generation.
- Build aggregation queries for manager dashboards.
- Add anonymized log access for administrator analytics.
- Test API calls using mock data.
- Optimize database queries.
- Implement security checks such as input validation and token verification.

Frontend Team (React, UI, Dashboards, Modules)

- Set up the React and TypeScript project structure.
- Add Google OAuth sign-in button and redirect flow.
- Build the landing page.

- Build the general user dashboard and metrics view.
- Add simulation history table and status indicators.
- Build the training module pages and layout.
- Implement module completion logic with GraphQL.
- Build the manager dashboard.
- Add user list, school statistics, and module assignment features.
- Add UI for simulation frequency settings.
- Add UI for manager join codes and invites.
- Implement responsive design for all screen sizes.
- Create charts and visualizations for analytics.
- Connect all pages to GraphQL queries and mutations.
- Integrate redirect-based training pages with backend logic.
- Polish the UI for consistency.
- Test UI on all major screen sizes.

Integration and Testing (All Teams)

- Test full workflow from login to email to training module.
- Confirm all database writes occur correctly.
- Test school creation and joining.
- Test manager permissions and access control.
- Test simulation scheduling with multiple users.
- Test browser compatibility.
- Test email rendering in common email clients.
- Load test email sending endpoints.
- Check analytics calculations for accuracy.
- Fix bugs found during testing.
- Perform final integrated system walkthrough

5.2 Schedule

Task Description	Assigned Members	Est Start	Est End	Actual Start	Actual End
Project Initialization & Architecture Setup	Victor Berrios, Milo Pumford, Max Olvey, Shane Norden, Sarah Smith, Paul Tribble	01/12	01/19		
Finalize Requirements, Roles, and Feature Scope	Paul Tribble, Victor Berrios	01/19	01/26		
Design Database Schema	Milo Pumford, Shane Norden	01/26	02/02		
Create Template Categories and Module Outlines	Victor Berrios	01/26	02/02		

Set Up Vercel Server Environment	Paul Tribble	02/02	02/09		
Set Up React Frontend	Sarah Smith	02/02	02/09		
Implement Google OAuth Login Flow	Max Olvey	02/02	02/16		
Build PostgreSQL Instance + Seed Data	Milo Pumford, Shane Norden	02/09	02/16		
Draft Full Training Module Content	Victor Berrios	02/09	02/23		
Write Phishing Email Templates	Victor Berrios	02/16	02/23		
Landing Page and Dashboard UI	Sarah Smith, Max Olvey	02/09	02/16		
Set Up GraphQL Server + Basic Queries	Paul Tribble	02/16	02/23		
GraphQL Mutations (logging, modules, settings)	Paul Tribble	02/23	03/02		
Backend → Frontend Integration	Sarah Smith, Paul Tribble	02/23	03/09		
SendGrid Email Integration	Paul Tribble	02/23	03/02		
Tokenized Tracking Links	Max Olvey, Paul Tribble	03/02	03/09		
Redirect Endpoint (lookup → module)	Paul Tribble	03/09	03/16		
Training Module Pages (Frontend)	Sarah Smith	03/02	03/16		
Manager Dashboard UI	Sarah Smith	03/16	03/30		
School Creation + Join Codes	Paul Tribble, Max Olvey	03/16	03/30		
Simulation Scheduler (frequency, random selection)	Paul Tribble	03/23	03/30		
Scheduler → Email Integration	Paul Tribble	03/30	04/06		
Analytics Queries (school stats, event logs)	Milo Pumford, Shane Norden	03/23	04/06		
Frontend Charts + Analytics Visuals	Sarah Smith	03/30	04/06		
End-to-End Integration Testing	Sarah Smith, Paul Tribble	04/06	04/13		

System Testing (multi-user, browser, email clients)	Entire Team	04/13	04/20		
Bug Fixing + UI Polish	Sarah Smith, Paul Tribble	04/20	04/27		
Finalize Templates & Modules	Victor Berrios	04/20	04/27		
Write Documentation (API, user guide, manager guide)	Entire Team	04/19	04/28		
Prepare Final Presentation & Demo	Entire Team	04/27	05/02		

5.3 Deliverables

The final deliverables for PhishWise will include a fully functional prototype of the phishing-education platform, a complete backend with a working PostgreSQL database, and a deployed web application. Because the long-term goal of PhishWise is to support large groups of users, the scope of this project is to deliver a fully working proof-of-concept system that demonstrates the complete simulation-training loop: generating phishing emails, tracking interactions, and redirecting users to appropriate training modules. A successful demo will show that emails can be sent, interactions are logged, user progress is tracked, and managers can view analytics through the dashboard.

In addition to the working software, a detailed documentation package will be included. This documentation will provide step-by-step instructions for setting up the development environment, configuring OAuth, deploying the system, and extending the training content. The document must be thorough to ensure that future developers, and potential adopters, can replicate the system, extend it, or deploy it independently without ambiguity.

The PostgreSQL database schema produced for this project will also be included as a deliverable. It will be designed to support future expansion, such as additional training modules, more advanced analytics, integration with other communication APIs, or more detailed simulation types. This schema will remain available for continued development beyond the initial prototype.

All frontend, backend, and integration code will be submitted as part of the project's final package. This includes the React application, GraphQL API, OAuth configuration, email dispatch logic, database code, and all supporting assets. Future developers will be able to build on top of these components to incorporate additional features or refine the existing ones. Alongside the code, all design tools used during development (such as Figma mockups or architectural diagrams) will be included to support further UI or backend enhancements.

Everything produced during this project will be accessible for future expansion, whether by sponsors, instructors, or future student teams. The system is designed to be extendable, open to additional features, and capable of supporting future improvements to phishing-awareness education.

6.0 Key Personnel

Sarah Smith – Smith is a senior in Computer Science in the Electrical Engineering and Computer Science Department at the University of Arkansas. Smith has experience working in industry as a past J.B. Hunt Transport Software Engineering Intern in Fayetteville, Arkansas. She worked in an agile cross-team setting to create a full-stack API. This was used to increase efficiency within the company, increase accessibility of the function, and create a mass adherence to NMFTA Laws. In this project, Smith will continue to develop her knowledge of user experience and front-end development.

Paul Tribble – Tribble is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. He has experience in software development, cybersecurity, and server setup. His past work includes building a phishing email detection model, running red-team security tests on HPC clusters, and creating an inventory management system used in university programs. For the PhishWise project, Paul is helping lead backend development, managing the Vercel server setup, and building the systems for sending and tracking phishing simulations. He's also working on API integration for email as well as helping design the database and training content features.

Milo Pumford – Pumford is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. For PhishWise, Pumford will help with background research and backend development.

Warren Olvey – Olvey is a senior Computer Engineering major in the Electrical Engineering and Computer Science Department at the University of Arkansas. He is experienced in software development and working with servers. He had an internship as a software developer for Arkansas Electric Cooperative Corporation and is currently working as a part time remote developer for them. Warren will be helping with the architecture of PhishWise, as well as the backend development of the project.

Victor Berrios – Berrios is a senior Computer Engineering major with a Cybersecurity concentration. He has had coursework in cybersecurity through Cryptography and Information Security. In addition, he has aided in designing over 100 hours of engineering curriculum for the University of Arkansas' engineering summer camps. For the PhishWise project, Berrios will assist in designing the cybersecurity training modules and help develop the user interface. His focus will be on improving the user experience and interactivity of the system's front-end training components.

Shane Norden – Norden is a senior Computer Science major in the Electrical Engineering and Computer Science Department at the University of Arkansas. He has experience in software development, database management systems, and cybersecurity. He will assist with backend development and server implementation.

7.0 Facilities and Equipment

Facilities and Equipment

PhishWise requires minimal hardware and relies on a small number of cloud services:

Development Machines

Team members use personal laptops/desktops. No additional hardware is required.

Vercel Pro - \$20/month

Used to host the frontend, backend, and database during development and testing.
Provides production-grade deployment, better performance, and higher usage limits.

SendGrid - Free Tier (up to 100 emails/day)

Used to send phishing simulation emails.

Paid plan available: Essentials starts at \$19.95/month if higher volume is needed.

Google Cloud OAuth - Free

Google Identity Services are free unless API quota limits are exceeded, which is unlikely for this project.

PostgreSQL Database - Free Tier

Using Vercel's integrated Postgres or another free-tier cloud Postgres provider.

Paid managed Postgres options start around \$9–\$15/month, but the free tier is sufficient.

Development Tools - Free

Node.js, React, TypeScript, GitHub, pgAdmin/TablePlus, Chrome DevTools, and Figma (free tier) will be used throughout development at no cost.

Testing Devices

Testing is performed on team laptops and personal mobile devices. No additional equipment required.

8.0 References

- [1] Grilli, Matthew D, et al. “Is This Phishing? Older Age Is Associated with Greater Difficulty Discriminating between Safe and Malicious Emails.” *The Journals of Gerontology: Series B*, vol. 76, no. 9, 30 Dec. 2020, pp. 1711–1715, <https://doi.org/10.1093/geronb/gbaa228>.
- [2] Phishing Threat Database, <https://cofense.com/knowledge-center-hub/real-phishing-email-examples>
- [3] Guide to phishing <https://www.valimail.com/resources/guides/guide-to-phishing/>
- [4] National Institute of Standards and Technology. “Phishing - Glossary | CSRC.” *Nist.gov*, 2015, csrc.nist.gov/glossary/term/phishing.
- [5] FBI IC3 2024 Report, https://www.ic3.gov/AnnualReport/Reports/2024_IC3Report.pdf
- [6] CanIPhish website, <https://caniphish.com/>
- [7] Microsoft Attack Simulation Training <https://www.microsoft.com/en-us/security/business/threat-protection/attack-simulation-training>
- [8] PhishingBox Phishing Test <https://www.phishingbox.com/phishing-test>
- [9] “The Latest Phishing Statistics (Updated June 2025): AAG IT Support.” *AAG IT Services*, 3 June 2025, aag-it.com/the-latest-phishing-statistics.

- [10] Gottfried, Jeffrey. "Online Scams and Attacks in America Today." *Pew Research Center*, Pew Research Center, 31 July 2025, www.pewresearch.org/internet/2025/07/31/online-scams-and-attacks-in-america-today/#:~:text=According%20to%20a%20Pew%20Research%20Center%20survey%2C,because%20of%20an%20online%20scam%20or%20attack**.
- [11] "Older Age Shown as Major Risk Factor in Falling for Phishing Scams" McKnight Brain Institute " University of Florida." *UF Monogram*, mbi.ufl.edu/2024/08/12/older-age-shown-as-major-risk-factor-in-falling-for-phishing-scams. Accessed 19 Oct. 2025.
- [12] Dymoke, Edward. "How Much Does Phishing Really Cost Businesses? - Hoxhunt." RSS, Hoxhunt, 13 Sept. 2024, hoxhunt.com/blog/how-much-does-phishing-really-cost-businesses#:~:text=of%20email%20cybersecurity.-,How%20the%20survey%20was%20conducted,total%20amounted%20to%20\$45,726%20annually.
- [13] "9 Examples of Social Engineering Attacks." *Terranova Security*, www.terranovasecurity.com/blog/examples-of-social-engineering-attacks. Accessed 19 Oct. 2025.
- [14] Knowbe4 Security Awareness Training Pricing." *KnowBe4 Security Awareness Training Pricing*, KnowBe4, www.knowbe4.com/products/security-awareness-training/pricing#:~:text=Monthly%20price,Get%20a%20quote. Accessed 19 Oct. 2025.