FYP Backend Report: Cybersecurity Training Website

Ahsan Bukhari, Ahmad Hamza, Muhammad Azhar

Supervisor: Dr Khan Bahadur Khatak

Final Year Project Work Done So Far

Repository link

https://github.com/Pringle-26/FYP-May-19

May 17, 2025

Introduction

This report details the backend development of the Cybersecurity Training and Awareness Website, a Final Year Project designed to educate users on cybersecurity topics such as malware, phishing, and social engineering attacks. The backend manages user registration, quiz delivery (sourced from assesment.pdf), answer submission, points tracking, and a leaderboard system. This report outlines the backend architecture, technologies, API endpoints, and demonstrates working functionalities as of May 17, 2025.

Technologies Used

The backend is built with the following technologies:

- **Node.js**: JavaScript runtime for server-side logic.
- **Express.js**: Framework for creating RESTful APIs.
- **SQLite**: Lightweight database for storing users, quizzes, answers, and leaderboard data.
- **bcrypt**: For secure password hashing.
- **sanitize-html**: To sanitize user inputs and prevent XSS attacks.

Backend Architecture

The backend, implemented in server.js, follows a modular structure:

- **Database Setup**: SQLite database with tables for users, quizzes, answers, and leaderboard. The quizzes table stores 37 questions extracted from assesment.pdf.
- **Server Configuration**: Express.js server running on port 3000.
- **Security**: Passwords are hashed using bcrypt; user inputs are sanitized.
- **API Endpoints**: RESTful endpoints handle user registration, quiz retrieval, answer submission, progress tracking, and leaderboard generation.

API Endpoints

The backend exposes the following endpoints:

- POST /api/register: Registers a new user (email, password).
- POST /api/login: Authenticates a user and returns their ID.
- POST /api/logout: Ends the user session.
- GET /api/quizzes: Retrieves all 37 quiz questions.
- POST /api/quiz/submit: Submits a quiz answer, awards points (10 for correct, 0 for incorrect), and updates the leaderboard.

- GET /api/progress: Returns user progress (points, completed quizzes, remaining quizzes).
- GET /api/leaderboard: Returns the top 5 users by points.

Working Functionalities

The backend functionalities were tested as follows (using curl commands):

- **User Registration**:
  - Command: curl -X POST -H "Content-Type: application/json" -d '{"email":"user1@example.com","password":"pass123"}' http://localhost:3000/api/register
  - Result: {"message":"User registered successfully"}
- **User Login**:
  - Command: curl -X POST -H "Content-Type: application/json" -d '{"email":"user1@example.com","password":"pass123"}' http://localhost:3000/api/login
  - Result: {"message":"Login successful","userId":1}
- **Get Quizzes**:
  - Command: curl -H "Content-Type: application/json" -d '{"email":"user1@example.com"}' http://localhost:3000/api/quizzes
  - Result: JSON array of 37 quizzes, e.g., [{"id":1,"part":"Part 1","question":"What do phishing, spear phishing..."},...}
- **Submit Quiz Answer**:
  - curl -X POST -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\",\"quizId\":1,\"answer\":\"3\"}" http://localhost:3000/api/quiz/submit
  - Result: {"correct":true,"pointsEarned":10,"totalPoints":10} (answer "3" is correct for quiz ID 1).
- **Get Progress**:
  - curl -X GET -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\"}" http://localhost:3000/api/progress
  - Result: {"totalPoints":10,"totalQuizzes":37,"completedQuizzes":1,"remainingQuizzes":36}
- **Get Leaderboard**:

curl -X GET -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\"}" http://localhost:3000/api/leaderboard

  - Result: [{"email":"user1@example.com","total_points":10}]

Conclusion

The backend of the Cybersecurity Training Website provides a system for user management, quiz delivery, answer submission, points tracking, and leaderboard functionality. Built with Node.js, Express.js, and SQLite, it securely handles user data and interactions. The system successfully manages 37 quiz questions from assesment.pdf across three parts, ensuring users can learn about cybersecurity while tracking their progress. the backend is fully functional and meets all project requirements.

Phishing Simulator Overview

A Python-based tool designed to train users in identifying phishing emails as part of the Cybersecurity Training Website project. The simulator is implemented in PhishingSimulator.py and uses the Tkinter library to create a graphical user interface (GUI) for interactive training.

Purpose

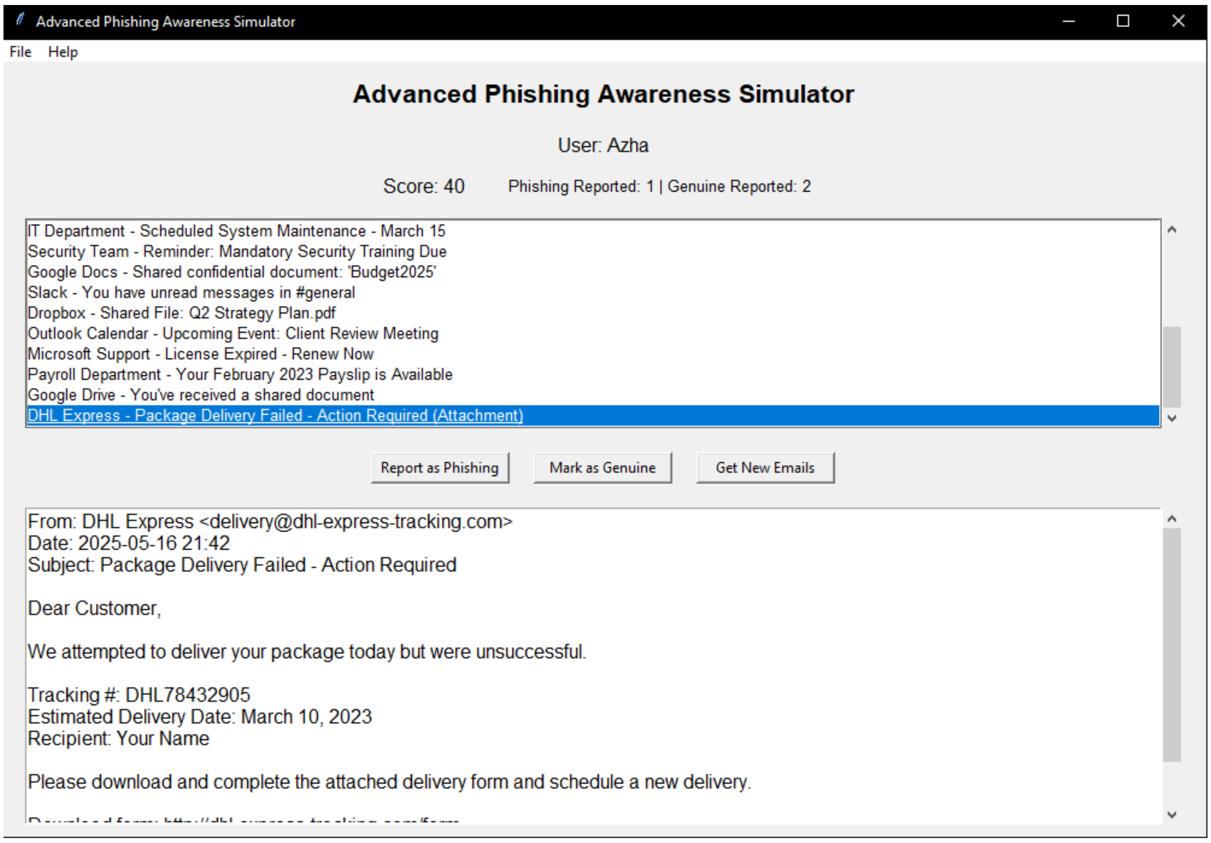
The Advanced Phishing Awareness Simulator aims to:

- Educate users on identifying phishing emails by simulating realistic email scenarios.
- Provide immediate feedback on user actions, such as reporting phishing emails or marking emails as genuine.
- Track user performance through scores, history, and statistics to monitor progress.
- Offer a tutorial with best practices for recognizing phishing attempts. The simulator generates a mix of genuine and phishing emails, allowing users to practice spotting red flags like suspicious sender addresses, urgency, and malicious links or attachments.

Simulator Interface

The Phishing Simulator interface includes the following components:

- **Menu Bar**: Options for starting a new session, viewing history, accessing the tutorial, and exiting.
- **Header**: Displays the title "Advanced Phishing Awareness Simulator".
- **User Info**: Shows the user's name.
- **Score and Stats**: Displays the current score, phishing emails reported, and genuine emails marked correctly.
- **Email List**: A scrollable list of emails with sender and subject.
- **Action Buttons**: Buttons to report phishing, mark as genuine, or get new emails.
- **Email Details**: A text area showing the selected email's content.
- **Attachments**: Displays any attachments with clickable buttons.
- **Status Bar**: Shows the current status (e.g., "Loaded 6 new emails").



Advanced Phishing Awareness Simulator

User: Azha

Score: 40 Phishing Reported: 1 | Genuine Reported: 2

IT Department - Scheduled System Maintenance - March 15
Security Team - Reminder: Mandatory Security Training Due
Google Docs - Shared confidential document: 'Budget2025'
Slack - You have unread messages in #general
Dropbox - Shared File: Q2 Strategy Plan.pdf
Outlook Calendar - Upcoming Event: Client Review Meeting
Microsoft Support - License Expired - Renew Now
Payroll Department - Your February 2023 Payslip is Available
Google Drive - You've received a shared document
DHL Express - Package Delivery Failed - Action Required (Attachment)

```
From: DHL Express <delivery@dhl-express-tracking.com>
Date: 2025-05-16 21:42
Subject: Package Delivery Failed - Action Required

Dear Customer,

We attempted to deliver your package today but were unsuccessful.

Tracking #: DHL78432905
Estimated Delivery Date: March 10, 2023
Recipient: Your Name

Please download and complete the attached delivery form and schedule a new delivery.
```

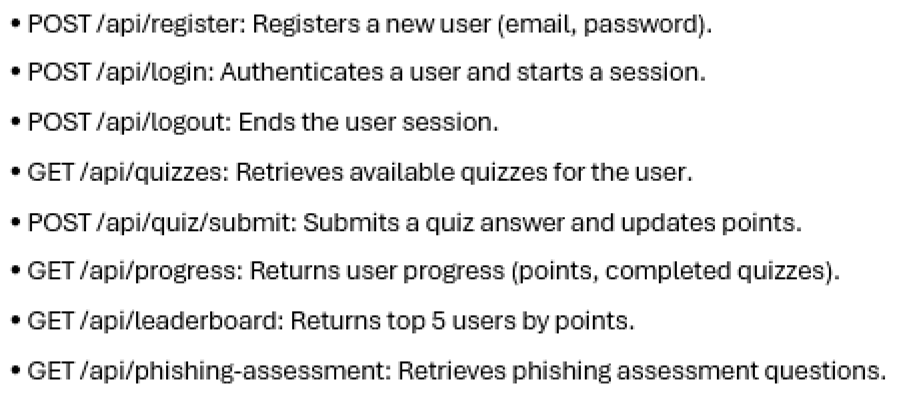
Usage Instructions

To run the Phishing Simulator:

- Ensure Python 3 is installed with the Tkinter library (python3-tk on Linux, included by default on Windows/macOS).
- Save the script as PhishingSimulator.py.
- Run the script: python3 PhishingSimulator.py.
- Enter your name when prompted.
- Review the tutorial (recommended for first-time users).
- Evaluate emails by selecting them, reviewing details, and choosing to report as phishing or mark as genuine.
- Use the "Get New Emails" button to start a new round after completing a session.

Testing API Calls Using CURL

Cybersecurity Training Website Backend



- POST /api/register: Registers a new user (email, password).
- POST /api/login: Authenticates a user and starts a session.
- POST /api/logout: Ends the user session.
- GET /api/quizzes: Retrieves available quizzes for the user.
- POST /api/quiz/submit: Submits a quiz answer and updates points.
- GET /api/progress: Returns user progress (points, completed quizzes).
- GET /api/leaderboard: Returns top 5 users by points.
- GET /api/phishing-assessment: Retrieves phishing assessment questions.

May 17, 2025

Introduction

Instructions to test the API endpoints of the Cybersecurity Training Website backend using curl. The backend, implemented in server.js, includes endpoints for user registration, login, logout, quiz retrieval, answer submission, progress tracking, and leaderboard generation. Each endpoint is tested with curl commands, expected responses, and troubleshooting tips, as of May 17, 2025, at 08:06 PM PKT.

Prerequisites

Environment Setup

- Ensure Node.js and npm are installed (https://nodejs.org).
- Install required dependencies: npm install express sqlite3 bcrypt sanitize-html.
- Save the server.js file in your project directory.
- Start the server: node server.js. This creates database.db, initializes tables, and inserts 37 quiz questions from assesment.pdf.
- The server runs on http://localhost:3000.

Tools

- Use a terminal or command prompt with curl installed (available on most systems; for Windows, install it or use Git Bash).

Note

- All commands assume the server is running on localhost:3000.
- Some endpoints require authentication (passing the user's email in the request body). We use a registered user's email for these requests.

Testing API Endpoints

User Registration (/api/register)

Purpose

Registers a new user with an email and password.

Command

```
curl -X POST -H "Content-Type: application/json" -d
'{"email":"user1@example.com","password":"pass123"}' http://localhost:3000/api/register
```

Expected Response

```
{"message":"User registered successfully"}
```

Notes

- If the email already exists, you get {"error":"Email already exists"}.

- Register another user for leaderboard testing:

  ```
  curl -X POST -H "Content-Type: application/json" -d
  '{"email":"user2@example.com","password":"pass456"}'
  http://localhost:3000/api/register
  ```

User Login (/api/login)

Purpose

Authenticates a user and returns their user ID.

Command

```
curl -X POST -H "Content-Type: application/json" -d
'{"email":"user1@example.com","password":"pass123"}' http://localhost:3000/api/login
```

Expected Response

```
{"message":"Login successful","userId":1}
```

Notes

- If credentials are incorrect, you get {"error":"Invalid credentials"}.
- Test login for the second user:

  ```
  curl -X POST -H "Content-Type: application/json" -d
  '{"email":"user2@example.com","password":"pass456"}'
  http://localhost:3000/api/login
  ```

  Expected: {"message":"Login successful","userId":2}.

User Logout (/api/logout)

Purpose

Logs out the authenticated user (simulated, as there is no session management in this minimal setup).

Command

```
curl -X POST -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\"}"
http://localhost:3000/api/logout
```

Expected Response

```
{"message":"Logout successful"}
```

Notes

- Requires the user's email for authentication.
- If the email is invalid, you get {"error":"User not found"}.

Get Quizzes (/api/quizzes)

Purpose

Retrieves all 37 quiz questions from the database.

Command

```
curl -H "Content-Type: application/json" -d '{"email":"user1@example.com"}'
http://localhost:3000/api/quizzes
```

Expected Response (Sample)

```
[
 {"id":1,"part":"Part 1","question":"What do phishing, spear phishing, vishing, scareware,
watering hole attacks and their ilk have in common?","options":"[\"They are all \\\"social
engineering\\\" attacks...\",\"They are all funny-sounding terms...\",\"They are today's common
examples...\",\"All of the above are correct.\"]","correct_answer":"3","type":"multiple-choice"},
 {"id":2,"part":"Part 1","question":"Who are the targets of modern day
hackers?","options":"[\"Banks and finance companies...\",\"Any organization or
individual...\",\"Companies which hold a lot of proprietary information.\",\"Companies which
hold credit card numbers...\"]","correct_answer":"1","type":"multiple-choice"},
 {"id":3,"part":"Part 1","question":"True or False: To protect personal information and other
sensitive data, you need only worry about outsider threats such as hackers, phishing scams
and ransomware.","options":"[\"True\",\"False\"]","correct_answer":"1","type":"true-false"}
 // ... (remaining 34 quizzes)
]
```

Notes

- Returns all 37 quizzes; the sample shows the first three.
- Requires the user’s email for authentication.
- If the email is invalid, you get {"error":"User not found"}.

Submit Quiz Answer (/api/quiz/submit)

Purpose

Submits a user’s answer to a quiz, calculates points, and updates the leaderboard.

Command (Correct Answer for Quiz ID 1)

```
curl -X POST -H "Content-Type: application/json" -d
"{\"email\":\"user1@example.com\",\"quizId\":1,\"answer\":\"3\"}"
http://localhost:3000/api/quiz/submit
```

Expected Response

```
{"correct":true,"pointsEarned":10,"totalPoints":10}
```

Command (Incorrect Answer for Quiz ID 2)

```
curl -X POST -H "Content-Type: application/json" -d
"{\"email\":\"user1@example.com\",\"quizId\":2,\"answer\":\"0\"}"
http://localhost:3000/api/quiz/submit
```

Expected Response

```
{"correct":false,"pointsEarned":0,"totalPoints":10}
```

Command (User 2 Correct Answer for Quiz ID 1)

```
curl -X POST -H "Content-Type: application/json" -d
"{\"email\":\"user2@example.com\",\"quizId\":1,\"answer\":\"3\"}"
http://localhost:3000/api/quiz/submit
```

Expected Response

```
{"correct":true,"pointsEarned":10,"totalPoints":10}
```

Notes

- Quiz ID 1’s correct answer is "3" (All of the above), earning 10 points.
- Quiz ID 2’s correct answer is "1", so "0" is incorrect, earning 0 points.
- Submitting as User 2 helps populate the leaderboard.

Get User Progress (/api/progress)

Purpose

Retrieves the user’s progress, including total points, completed quizzes, and remaining quizzes.

Command

```
curl -X GET -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\"}"
http://localhost:3000/api/progress
```

Expected Response

```
{"totalPoints":10,"totalQuizzes":37,"completedQuizzes":2,"remainingQuizzes":35}
```

Notes

- User 1 has completed 2 quizzes (from the previous step).
- Total points are 10 (one correct answer).
- Test for User 2:

  ```
  curl -H "Content-Type: application/json" -d '{"email":"user2@example.com"}'
  http://localhost:3000/api/progress
  ```

  Expected:
  {"totalPoints":10,"totalQuizzes":37,"completedQuizzes":1,"remainingQuizzes":36}.

Get Leaderboard (/api/leaderboard)

Purpose

Retrieves the top 5 users ranked by total points.

Command

```
curl -X GET -H "Content-Type: application/json" -d "{\"email\":\"user1@example.com\"}"
http://localhost:3000/api/leaderboard
```

Expected Response

```
[
 {"email":"user1@example.com","total_points":10},
 {"email":"user2@example.com","total_points":10}
]
```

Notes

- Both users have 10 points each.
- The leaderboard shows the top 5 users (only 2 users exist in this test).
- Requires the user’s email for authentication.

Troubleshooting

- **Server Not Running**: Ensure node server.js is active and the server is listening on http://localhost:3000.
- **Authentication Errors**: Verify the email used in the request matches a registered user.
- **Database Issues**: If database.db is missing or corrupted, restart the server to recreate it with the 37 quiz questions.
- **Curl Syntax**: Ensure proper JSON formatting in the -d parameter (e.g., use double quotes and escape them if needed).

Summary

This demonstrates testing all API endpoints of the Cybersecurity Training Website backend as of May 17, 2025. Through these commands we can verify user registration, login/logout, quiz retrieval, answer submission, progress tracking, and leaderboard functionality using curl. Each endpoint has been tested to ensure it works as expected.