Solution Architecture Document (SAD)

Section 1: Project Objectives

The goal of this project is to design and build an interactive and realistic cybersecurity training game which educates employees about security awareness.

- What problems are you solving?
  1. Employees often lack practical, engaging cybersecurity training.
  2. Traditional training methods (e.g., slide decks, generic videos) are not interactive, leading to low retention.
- Who is the target audience?
  1. Company employees across all departments.
  2. IT Students looking to enhance their cybersecurity knowledge
- What outcomes should this solution achieve?
  1. improve decision-making during real-world cyber incidents.
  2. Reduce loss made when an attack is involved.
  3. increase employee awareness of cybersecurity threats.

Section 2: Business Requirements

User Stories (Use format below):

As an employee,

I need to be able to practice identifying phishing emails by playing in a realistic scenario where I am sent different emails and I should be able to identify which could possibly be phishing emails.

so that I can avoid falling victim to real phishing attempts.

As an IT support staff,

I need to be able to respond to simulated ransomware attacks by choosing through a decision tree with multiple different answers that would affect the final score or ending of the storyline,

so that I can practice quick containment and communication procedures.

As an HR staff member,

I need to be able to recognize social engineering attempts via phone or chat by playing in realistic scenarios of phone calls or chats sent to me and recognizing the abnormal calls,

so that I can protect sensitive employee data.

// Todo: Add in user story for security admin

Other crypto security scenarios:
Phishing for Wallet Keys:A fake email or Discord message pretends to be from a major exchange ("Verify your MetaMask now or funds will be locked!").

Impersonation in Telegram/Discord:An attacker poses as a **project admin**: "We're doing emergency maintenance—send funds to this wallet for migration."

Acceptance Criteria:

- Players are able to identify 90% of the phishing tests.
- Players make the correct or near correct decisions for at least 3 of the simulations

Section 3: Technical Requirements

Technology Stack:

- Frontend:React.js
- Backend:Node.js
- Database:MongoDB
- AI/Data Tools (if any):ChatGPT
- APIs/Integrations:

User Story Implementation Mapping:

User Story	Implementation Approach	Functions/Components Required
IT support staff	Incident Response Game Mode	Timed gameplay with backend random event triggers and scoring
Hr Staff	Social Engineering Scenarios	Interactive chat/voice simulation with branching choices
employee	Phishing Simulation Module	Frontend email client UI + backend phishing detection scorin

(Repeat for each major feature or user story)

Section 4: Data Model

1) User

- `id` (uuid)
- `email` (string)
- `name` (string)
- `role` (enum: employee | manager | admin)

- `created_at` (timestamp)

2) AssessmentResult

- `id` (uuid)
- `assessment_id` (uuid)
- `user_id` (uuid)
- `score` (int)
- `taken_at` (timestamp)

3) Scenario

- `id` (uuid)
- `title` (string)
- `category` (enum: phishing | passwords | social | ransomware | incident)
- `difficulty` (enum: beginner | intermediate | advanced)
- `is_active` (bool)

4)Assessment

- `id` (uuid)
- `org_id` (uuid)
- `name` (string)
- `kind` (enum: pre | post)
- `items` (json)

5) Choice

- `id` (uuid)
- `step_id` (uuid)
- `label` (string)
- `next_step_id` (uuid, nullable)
- `score_delta` (int)
- `feedback` (text, short)

6) Playthrough

- `id` (uuid)
- `user_id` (uuid)
- `scenario_id` (uuid)
- `status` (enum: started | completed | abandoned)
- `score_total` (int)
- `time_spent_sec` (int)

- `started_at`, `completed_at` (timestamps)

7) Event (audit of actions)

- `id` (uuid)
- `playthrough_id` (uuid)
- `step_id` (uuid)
- `choice_id` (uuid, nullable)
- `type` (enum: view | choose | timeout | hint)
- `created_at` (timestamp)

Section 5: Diagrams

System Architecture Diagram:

- Overview of system components and their relationships (include frontend, backend, database, APIs, etc.)

Data Flow Diagram (DFD):

- Describe how data moves through the system across components.
- Use UML where appropriate.