Hasthiya IT
Intern Software Engineer Technical Assessment

2025 December

+94 71 256 0246 | developerhit@outlook.com

Assignment: Web Game Application ("Hasthiya Lucky 4")

Duration: 10 Hours

Role: Software Engineer Intern

Tech Stack: MERN (MySQL, Express, React, Node) with TypeScript & Vite

1. The Objective

You are tasked with building a small, interactive web game called **"Hasthiya Lucky 4"**. The goal is to test your ability to handle logic implementation, state management, database interactions, and pagination.

2. Technical Requirements

You **must** strictly use the following technology stack. Deviating from this stack will result in disqualification.

- **Frontend:** Vite + React + TypeScript + Tailwind CSS
- **Backend:** Node.js + Express.js
- **Database:** MySQL

3. Functional Requirements (The Task)

Your application must include the following features:

A. Game Setup (Backend Configuration)

- Define **4 Hidden Numbers** on your backend (e.g., in a constant file or .env).
- Each hidden number must be an integer between **1 and 10**.
- *Example Hidden Set:* [2, 5, 8, 3]

B. User Entry (No Auth, Unique Access)

1. **Welcome Screen:** A simple form asking for the user's **Email Address**.
2. **Validation:**

- Check if this email has **already played** the game.
- If the email exists in the database with a status of played: true, show an error: *"Game already played with this email."*
- If the email is new, allow them to proceed to the Game Screen.

C. The Gameplay

1. **Game Screen:** Display 4 distinct buttons (e.g., "Roll 1", "Roll 2", "Roll 3", "Roll 4").
2. **Action:** When a user clicks a button, it must generate a random number between **1-10** and display it.
3. **Completion:** Once all 4 numbers are generated, the system must calculate the score and automatically save the result.

D. Score Calculation (The Equation)

You must implement the following equation in your backend to determine the score:

$$Score = 100 - (\sum_{i=1}^{4} |\text{Hidden}_i - \text{Generated}_i|) \times 2$$

- *Explanation:* The score is 100 minus the sum of the absolute differences between the hidden numbers and user's numbers, multiplied by 2.
- *Minimum Score:* If the result is negative, store it as 0.

E. Leaderboard & Pagination

1. **Leaderboard Page:** Display a table of results.
2. **Sorting:** Records must be sorted by **Score** in Descending order (Highest score first).
3. **Pagination:** Display only **10 records per page**.
4. **Columns:** Rank, Email, Score, Date Played.

F. Database Schema

You may use the following SQL structure as a baseline:

```
CREATE TABLE game_results (
    id INT AUTO_INCREMENT PRIMARY KEY,
    email VARCHAR(255) NOT NULL UNIQUE,
```

```
    generated_numbers JSON, -- Stores the array like "[1, 4, 6, 2]"
    score INT DEFAULT 0,
    is_played BOOLEAN DEFAULT FALSE,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

C. Creativity (Bonus)

- You are encouraged to use your own creative ideas for the UI/UX design.

4. Repository & Directory Structure

Strict adherence to file and folder structure is required.

1. **Repository Setup:**
    - Create a **Public** GitHub repository in your personal account.
    - **Repository Name:** <YourName>-HasthiyaLuckyGame (e.g., SamanPerera-HasthiyaLuckyGame).
2. **Directory Structure:**
    - The root of the repository must contain exactly two folders:
        - frontend/ (All Vite/React code goes here)
        - backend/ (All Node/Express code goes here)

5. Version Control & Timeline Rules

We evaluate your workflow, not just the final code.

1. **Total Duration:** 10 Hours.
2. **Commit Frequency:**
    - You must push a commit to the repository **at least every 2 hours**.
    - *Note:* We will check the timestamps. Large dumps of code at the end are not acceptable.
    - We will **only** evaluate commits made during the assignment time window.
3. **Commit Messages:**
    - Messages must be descriptive and clear.
    - *Bad:* "fix", "update", "code"

- *Better:* "feat: create project add form with validation", "fix: resolve db connection error"

6. Evaluation Criteria

Your submission will be scored based on the following:

Category	Criteria
Logic & Correctness	Accurate implementation of the **Score Equation** and Random Number generation.
Code Structure	Clean architecture, separation of concerns (Controllers, Routes, Components), and file organization.
Best Practices	Proper naming conventions (camelCase, PascalCase), usage of TypeScript interfaces (no any), and modular code.
Comments	Code must be well-commented explaining complex logic.
Version Control	Adherence to the 2-hour commit rule and clarity of commit messages.
Functionality	The application runs without errors and meets all functional requirements.
Validations	Proper frontend and backend validation (e.g., empty fields, invalid dates).
UI/UX	Responsiveness, use of Tailwind CSS, and overall visual appeal.
AI Usage	**Strict Policy:** Using AI generators (ChatGPT, Copilot, etc.) to write bulk code will be detected. If detected, marks will be deducted significantly.

7. Submission Guidelines

Once the 10-hour duration is complete:

1. Ensure your repository is **Public**.
2. Submit the repository link using the form provided to you.

   https://forms.gle/yPy4qBcCfuKVNeXN9
3. If you have additional files (e.g., database exports, environment variable examples, screenshots), compress them into a .zip file and attach it to the form.

Good Luck! We look forward to seeing your best work.

For further clarification:- *Drop a text to +94779184997*