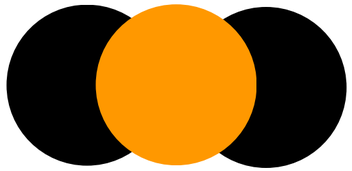
MindQuest Trivia Game

By Kipchumba Brian



Overview of MindQuest

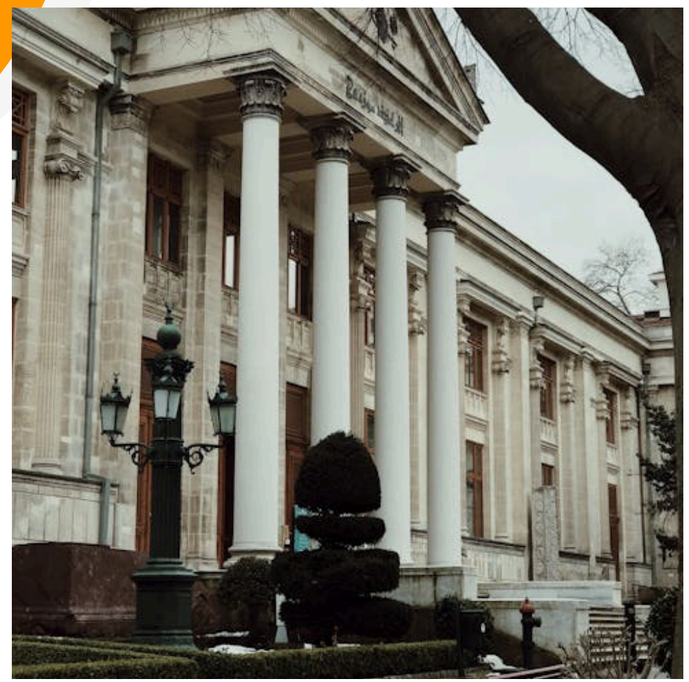
01 Interactive, fast-paced trivia game.

02 Built as a Single Page Application (SPA).

03 Fetches questions dynamically from Open Trivia Database API.

04 Seamless UI with no page reloads.

05 Developed using JavaScript, HTML, and CSS.



Key Development Practices

Handling Data & API Requests

MindQuest dynamically fetches trivia questions from an external API, ensuring a diverse set of categories and difficulty levels.

Using Asynchronous JavaScript

API calls, real-time user interactions, and data updates are managed using async/await, ensuring a smooth experience.

Frontend UI Structuring

The modular UI design allows for easy maintenance, enabling efficient updates and better user experience.

Core Features (MVP)

Custom Quiz Settings

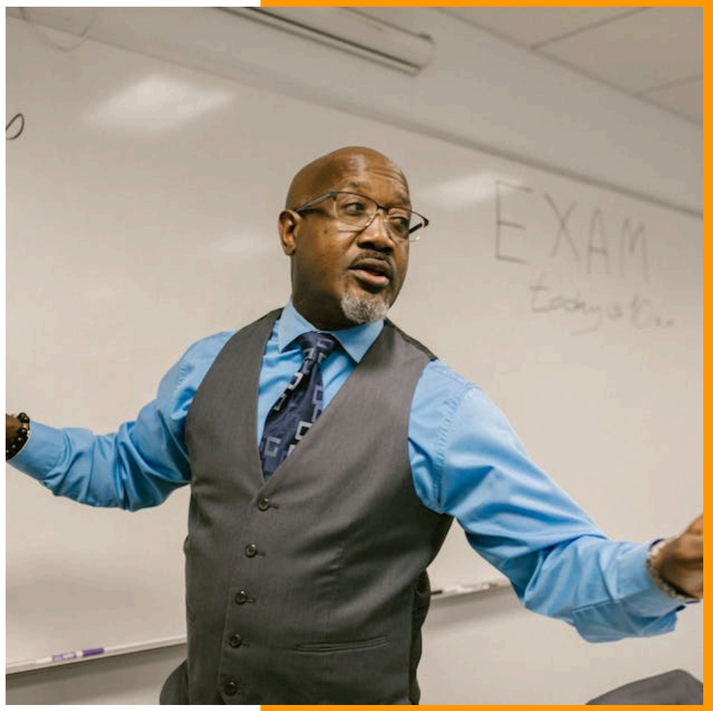
Users can select the number of questions to answer

Dynamic Question Fetching

Trivia questions are retrieved in real-time from the Open Trivia Database API, ensuring variety.

Real-Time User Interaction

Interactive elements like click events, answer submissions, and countdown timers enhance engagement.



Additional Features (Stretch Goals)

Shuffle Answers Function

To maintain fairness, the answer order is randomized for each question.

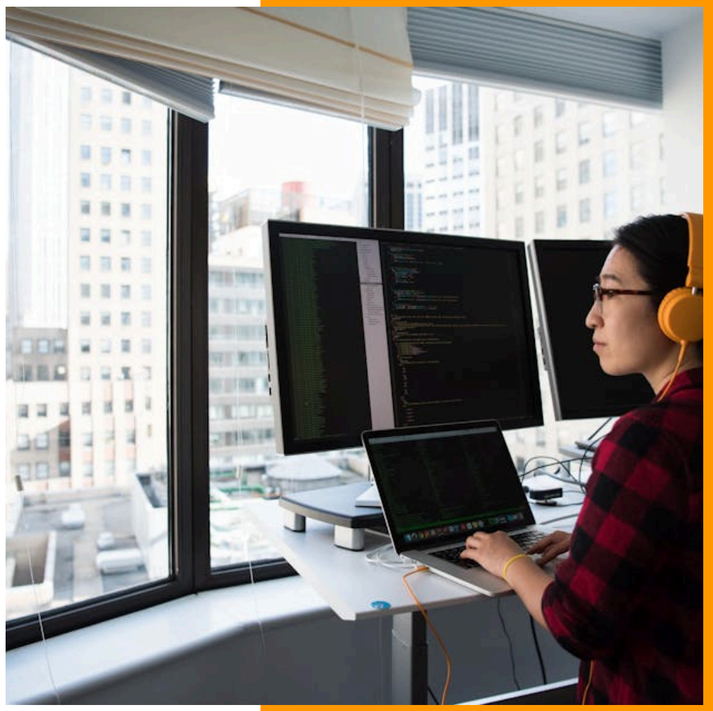
Animated UI Effects

Correct and incorrect answers are highlighted with animations for a better user experience.

Live Scoreboard & Restart Controls

Users can track their progress in real-time and restart the quiz whenever they choose.

Anticipated & Overcome Challenges



Handling API Errors

Implemented robust error handling to manage failed API requests and ensure a seamless user experience.

Optimizing Performance

Utilized efficient DOM manipulation and event delegation to enhance app responsiveness.

Ensuring Smooth Gameplay

Fixed navigation bugs and improved UI responsiveness to provide uninterrupted gameplay.

Future Enhancements (Next Steps)

Multiplayer Mode

Introducing a real-time multiplayer feature where users can compete against each other.

Leaderboard & User-Generated Quizzes

Allowing players to track high scores and create/share their own quiz questions.

Enhanced UX with Sound & Animations

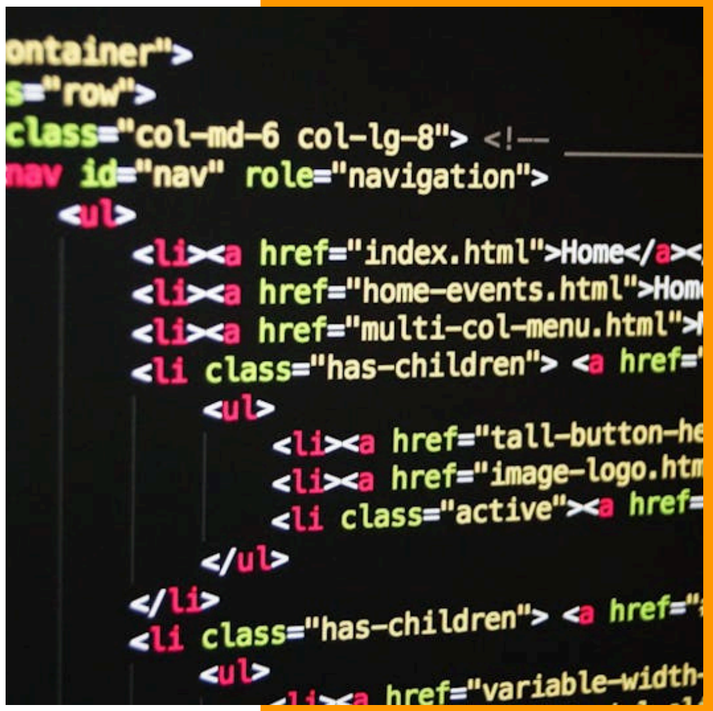
Implementing immersive sound effects and additional animations to improve engagement.

Table of Contents

Introduction

What is MindQuest Trivia Game?

MindQuest Trivia Game is an interactive web-based quiz application that allows users to test their knowledge through multiple-choice questions.



Purpose

The game provides an engaging way for users to challenge themselves, improve their knowledge, and enjoy a fun and interactive learning experience.

Key Features

- Users select the number of questions.
- Questions are fetched from an API.
- Navigation through questions with interactive UI.
- Results are displayed at the end, showing correct and incorrect answers.

Website Overview

How It Works

The MindQuest Trivia Game allows users to select the number of trivia questions they want to answer. The game dynamically fetches trivia questions from an API.

User Interaction

Users can navigate through questions using 'Next' and 'Previous' buttons. They select an answer and receive instant feedback after completing the quiz.

Result Display

Once all questions are answered, the game displays results, showing correct and incorrect responses along with the total score.

HTML Structure

Container

The container element holds all game components, including questions, answers, navigation buttons, and the result display.

Question Input

Users enter the number of questions they want to answer. This input determines the number of questions fetched from the API.

Question Display & Navigation

The game dynamically displays questions and answer choices. Navigation buttons allow users to move forward, backward, or restart the quiz.

CSS Styling

User Interface Design

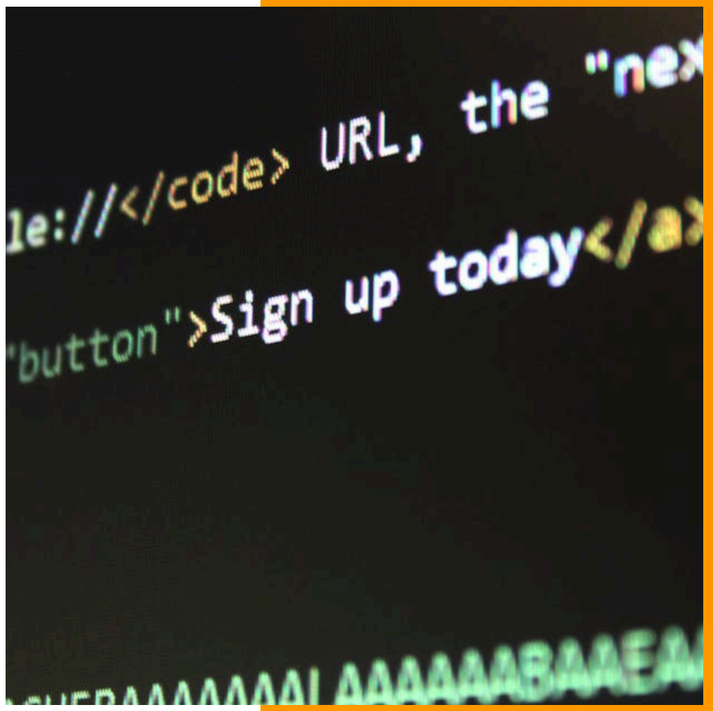
CSS styles are applied to enhance the visual appearance of the game, making it engaging and user-friendly. Styling includes layout, color schemes, and font choices.

Responsiveness

The game is designed to work across different screen sizes, ensuring an optimal experience on desktops, tablets, and mobile devices.

Animations & Effects

CSS transitions and animations enhance interactivity, providing visual feedback when users select an answer or navigate through questions.



JavaScript Functionality

Fetching Questions

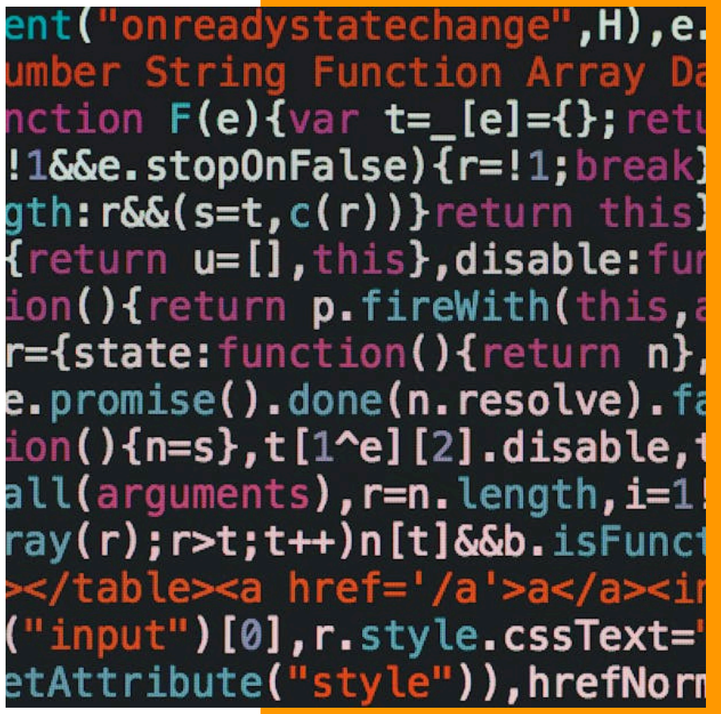
The game dynamically retrieves trivia questions from an external API based on user input. The API fetch is handled using JavaScript's Fetch API.

Handling User Answers

User responses are stored in an array and checked for correctness against the correct answer provided by the API.

Game Logic

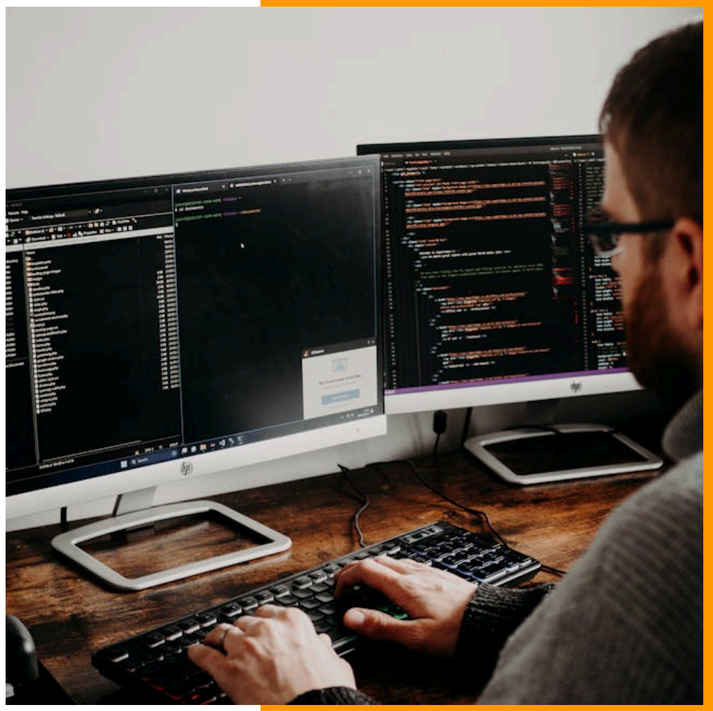
JavaScript functions control the game flow, including navigating through questions, tracking the score, and displaying results at the end.



Function: startGame()

Purpose

The `startGame()` function initializes the trivia game by fetching the number of questions input by the user and retrieving them from the API.



How It Works

1. Reads the number of questions entered by the user.
2. Makes an API request to fetch the trivia questions.
3. Sets up the initial question display and enables navigation.

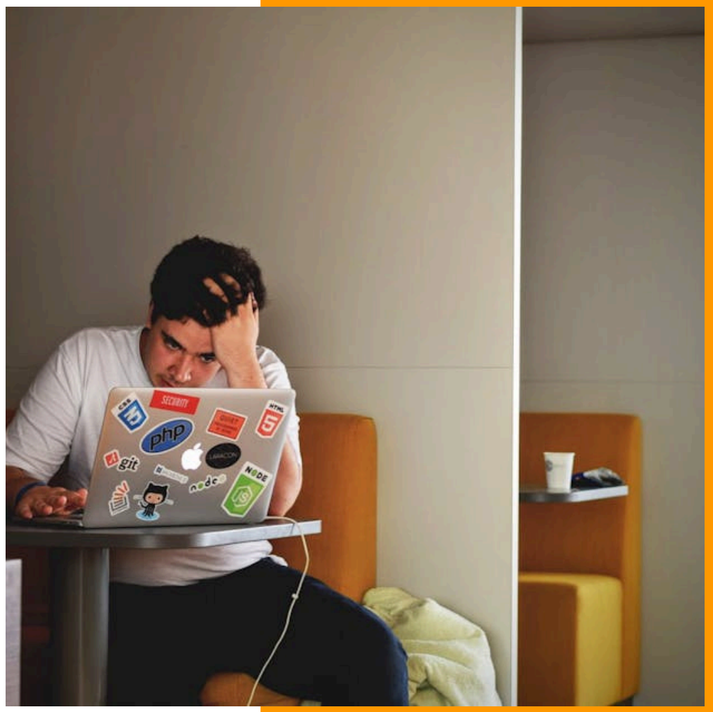
Key Features

- Ensures valid input before starting.
- Fetches only the required number of questions.
- Prepares the game UI for user interaction.

JavaScript Functions Overview

What are JavaScript Functions?

Functions in JavaScript are reusable blocks of code that perform a specific task. They help keep code organized and efficient.



How Functions Work

A function is defined using the `function` keyword followed by a name and parentheses. Inside the curly braces `{}` is the code to be executed.

MindQuest Trivia Functions

Several JavaScript functions power the MindQuest Trivia Game, including fetching questions, displaying content, handling user input, and calculating results.

fetchQuestions(amount) Function

Purpose

The `fetchQuestions(amount)` function is responsible for retrieving trivia questions from the Open Trivia Database API based on the user's selection.

Key Features

- Ensures input validation (1-20 questions)
- Fetches data dynamically using `fetch()`
- Randomizes answer order for fairness.

Example Implementation

Uses `async/await` to handle API responses and error handling.

displayQuestion() Function

Purpose

The `displayQuestion()` function dynamically presents the current question along with multiple-choice answers to the user.

Key Features

- Extracts question data from fetched API responses.
- Updates the DOM elements to display the question.
- Ensures selected answers are visually highlighted.

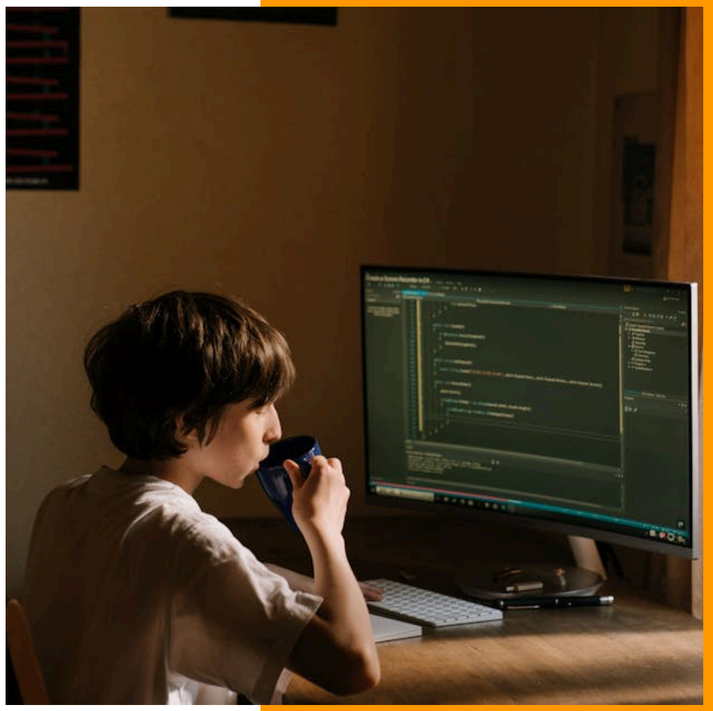
Example Implementation

Uses JavaScript event listeners to detect user selections and update styles accordingly.

showResults() Function

Purpose

The `showResults()` function calculates the user’s final score and displays the correct and incorrect answers after the quiz is completed.



Key Features

- Iterates through user-selected answers and compares them to correct answers.
- Dynamically updates the result section in the DOM.
- Displays a restart button for users to retry the quiz.

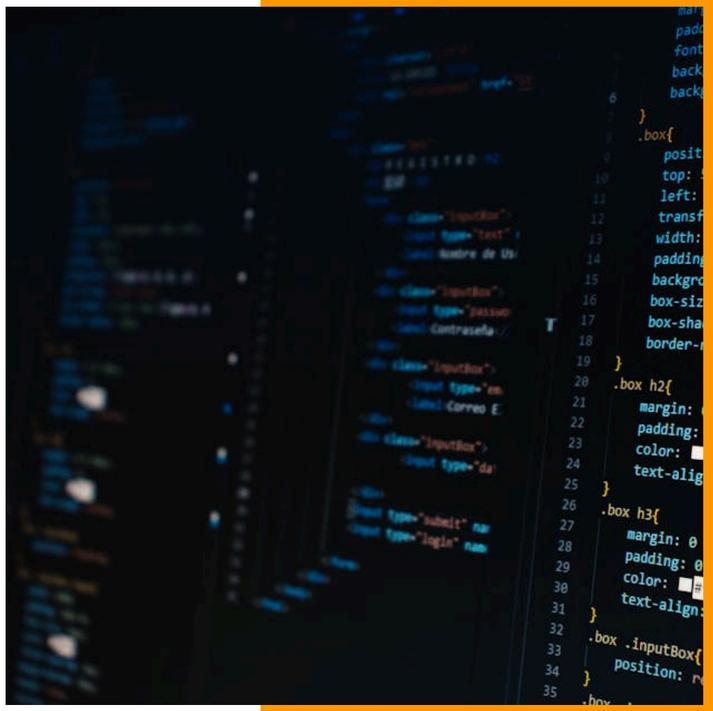
Example Implementation

Uses JavaScript loops and conditional logic to evaluate the user's performance and update the interface.

shuffleAnswers(answers) Function

Purpose

The `shuffleAnswers(answers)` function ensures that the order of answer choices is randomized to maintain fairness in the quiz.



Key Features

- Implements the Fisher-Yates shuffle algorithm.
- Randomly rearranges answer choices for each question.
- Prevents users from memorizing answer positions.

Example Implementation

Uses JavaScript array methods such as `.sort()` or `.splice()` to shuffle answer choices.