**Create admin.js to be accessed with localhost:3000/admin**

It builds a dynamic quiz admin dashboard that:

Loads quiz questions from an API

Renders each question with editable fields (text, choices, answer)

Allows deleting questions

Lets the admin add new questions

Saves the entire quiz to the backend

**Step 1: Initialize the data structure**

let quizData = [];

**Explanation:**

Creates an empty array to hold quiz questions from the server.

Used globally so it can be read/written by multiple functions.

**Step 2: Load data from API**

fetch('/api/quiz')

.then(res => res.json())

.then(data => {

quizData = data;

renderQuizAdmin();

})

.catch(err => {

console.error('Failed to load quiz:', err);

});

**Explanation:**

Fetches the quiz JSON from your backend.

Parses the response into JavaScript objects.

Stores them in quizData.

Calls renderQuizAdmin() to show the interface.

Logs errors to the console if something goes wrong.

**Step 3: Define renderQuizAdmin() function**

function renderQuizAdmin() {

const listDiv = document.getElementById('quiz-admin-list');

listDiv.innerHTML = '';

**Explanation:**

Gets the container where all editable questions will be shown.

Clears it out before rerendering.

**Step 4: Loop through quiz questions**

quizData.forEach((q, index) => {

const div = document.createElement('div');

div.className = 'border p-3 mb-3';

**Explanation:**

Loops over each question in quizData.

Creates a new div for each question, styled with Bootstrap.

**Step 5: Create editable input for question text**

const questionInput = document.createElement('input');

questionInput.type = 'text';

questionInput.className = 'form-control mb-2';

questionInput.value = q.question;

questionInput.oninput = () => {

quizData[index].question = questionInput.value;

};

div.appendChild(questionInput);

**Explanation:**

Creates a text input field.

Pre-populates it with the current question.

Updates quizData in real time as the user types.

**Step 6: Create editable input for choices**

const choicesInput = document.createElement('input');

choicesInput.type = 'text';

choicesInput.className = 'form-control mb-2';

choicesInput.value = q.choices.join(', ');

choicesInput.oninput = () => {

quizData[index].choices = choicesInput.value.split(',').map(c => c.trim());

};

div.appendChild(choicesInput);

**Explanation:**

Shows choices in one comma-separated string.

On input, splits the string back into an array.

Trims whitespace and updates quizData.

**Step 7: Create editable input for answer**

const answerInput = document.createElement('input');

answerInput.type = 'text';

answerInput.className = 'form-control mb-2';

answerInput.value = q.answer;

answerInput.oninput = () => {

quizData[index].answer = answerInput.value.trim();

};

div.appendChild(answerInput);

**Explanation:**

Editable field for the correct answer.

Trims whitespace and saves to quizData.

**Step 8: Add a delete button**

const delBtn = document.createElement('button');

delBtn.className = 'btn btn-danger btn-sm';

delBtn.textContent = 'Delete';

delBtn.onclick = () => {

quizData.splice(index, 1);

renderQuizAdmin();

};

div.appendChild(delBtn);

**Explanation:**

Deletes the current question from quizData.

Re-renders the entire list immediately after deletion.

**Step 9: Append each question to the list**

listDiv.appendChild(div);

**Explanation:**

Adds the entire block (question + inputs + button) to the page.

**Step 10: Add "Save All Changes" button**

const saveBtn = document.createElement('button');

saveBtn.textContent = 'Save All Changes';

saveBtn.className = 'btn btn-primary mt-3';

saveBtn.onclick = () => {

fetch('/api/quiz', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify(quizData)

})

.then(res => res.json())

.then(data => {

alert(data.message || 'Saved!');

})

.catch(err => {

console.error('Save failed:', err);

alert('Failed to save changes.');

});

};

listDiv.appendChild(saveBtn);

**Explanation:**

Adds a button to send the entire updated quiz to the backend.

Uses POST to update data on the server.

Alerts the user when saving is complete or if it fails.

**Step 11: Handle the “Add New Question” form**

document.getElementById('add-question-form').addEventListener('submit', function (e) {

e.preventDefault();

**Explanation:**

Prevents the page from reloading on form submit.

**Step 12: Read form values**

const question = document.getElementById('new-question').value.trim();

const choicesInput = document.getElementById('new-choices').value.trim();

const answer = document.getElementById('new-answer').value.trim();

const choices = choicesInput.split(',').map(c => c.trim()).filter(c => c);

**Explanation:**

Reads the values from the inputs.

Converts choices into an array of strings.

Removes empty values with filter().

**Step 13: Validate and push new question**

if (!question || choices.length < 2 || !answer) {

alert('Please enter a question, at least 2 choices, and an answer.');

return;

}

quizData.push({

question,

choices,

answer

});

**Explanation:**

Ensures each new question has:

A non-empty question

At least 2 choices

A valid answer

Adds the new object to quizData.

**Step 14: Rerender and reset form**

renderQuizAdmin();

this.reset();

});

**Explanation:**

Re-renders the admin list with the new question.

Clears the form inputs for the next entry.