AI Checker

Here’s a complete solution for your project, with a basic web interface using HTML/CSS/JavaScript and a Flask backend to manage question creation, answer submission, and automatic answer checking. This code will include MongoDB integration for question storage and the use of AI models for similarity scoring.

**Project Structure**

graphql

Copy code

question\_checker\_project/

├── app.py # Flask backend

├── templates/

│ ├── index.html # Frontend for question input

│ ├── submit.html # Frontend for answer submission

├── static/

│ ├── style.css # CSS styling for HTML pages

└── requirements.txt # Dependencies

**Step 1: Setting Up the Environment**

1. **Install Dependencies**:

bash

Copy code

pip install Flask transformers sentence-transformers pymongo

1. **requirements.txt**:

plaintext

Copy code

Flask

transformers

sentence-transformers

pymongo

1. **MongoDB Setup**: Make sure MongoDB is installed and running. Create a database named question\_checker and a collection named questions.

**Step 2: Backend (app.py)**

Create the backend with Flask to handle question creation and answer checking.

python

Copy code

from flask import Flask, render\_template, request, redirect, url\_for, jsonify

from sentence\_transformers import SentenceTransformer, util

from pymongo import MongoClient

from bson.objectid import ObjectId

app = Flask(\_\_name\_\_)

model = SentenceTransformer('paraphrase-MiniLM-L6-v2')

client = MongoClient('mongodb://localhost:27017/')

db = client['question\_checker']

questions\_collection = db['questions']

# Home page for adding questions

@app.route('/')

def home():

return render\_template('index.html')

# Endpoint to create a question

@app.route('/create\_question', methods=['POST'])

def create\_question():

question = request.form['question']

model\_answer = request.form['model\_answer']

questions\_collection.insert\_one({'question': question, 'model\_answer': model\_answer})

return redirect(url\_for('home'))

# Page for students to submit answers

@app.route('/submit')

def submit\_page():

questions = list(questions\_collection.find())

return render\_template('submit.html', questions=questions)

# Endpoint to check a student's answer

@app.route('/check\_answer', methods=['POST'])

def check\_answer():

question\_id = request.form['question\_id']

student\_answer = request.form['student\_answer']

question = questions\_collection.find\_one({'\_id': ObjectId(question\_id)})

model\_answer = question['model\_answer']

# Calculate similarity

embeddings = model.encode([model\_answer, student\_answer])

similarity = util.cos\_sim(embeddings[0], embeddings[1]).item()

# Determine result

result = "Correct" if similarity > 0.75 else "Incorrect"

return jsonify({

"question": question['question'],

"student\_answer": student\_answer,

"similarity": round(similarity, 2),

"result": result

})

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**Step 3: Frontend Templates**

**1. index.html – Question Input for Teachers**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Question Checker - Create Question</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

</head>

<body>

<h2>Create a New Question</h2>

<form action="/create\_question" method="POST">

<label for="question">Question:</label>

<input type="text" id="question" name="question" required>

<label for="model\_answer">Model Answer:</label>

<textarea id="model\_answer" name="model\_answer" required></textarea>

<button type="submit">Add Question</button>

</form>

<a href="{{ url\_for('submit\_page') }}">Go to Answer Submission</a>

</body>

</html>

**2. submit.html – Answer Submission for Students**

html

Copy code

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Submit Answer</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='style.css') }}">

<script>

async function checkAnswer() {

const questionId = document.getElementById('question\_id').value;

const studentAnswer = document.getElementById('student\_answer').value;

const response = await fetch('/check\_answer', {

method: 'POST',

headers: { 'Content-Type': 'application/x-www-form-urlencoded' },

body: `question\_id=${questionId}&student\_answer=${studentAnswer}`

});

const result = await response.json();

document.getElementById('result').innerText =

`Result: ${result.result}, Similarity: ${result.similarity}`;

}

</script>

</head>

<body>

<h2>Submit Your Answer</h2>

<label for="question\_id">Select Question:</label>

<select id="question\_id">

{% for question in questions %}

<option value="{{ question.\_id }}">{{ question.question }}</option>

{% endfor %}

</select>

<label for="student\_answer">Your Answer:</label>

<textarea id="student\_answer" required></textarea>

<button onclick="checkAnswer()">Check Answer</button>

<div id="result"></div>

</body>

</html>

**3. style.css – Basic Styling**

css

Copy code

body {

font-family: Arial, sans-serif;

margin: 20px;

max-width: 600px;

}

h2 {

color: #333;

}

form, label, select, textarea, input, button {

display: block;

width: 100%;

margin-top: 10px;

}

button {

padding: 10px;

background-color: #007BFF;

color: white;

border: none;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

#result {

margin-top: 20px;

font-weight: bold;

}

**Step 4: Running the Project**

1. **Start MongoDB** (if it’s not running already):

bash

Copy code

mongod

1. **Run the Flask App**:

bash

Copy code

python app.py

1. **Open the Web Interface**:
   * Go to http://127.0.0.1:5000/ in your browser to add questions.
   * Navigate to http://127.0.0.1:5000/submit for the student answer submission page.

**Explanation of Workflow**

* Teachers can add questions with model answers via the / page.
* Students can select a question and submit their answers on the /submit page.
* The backend API checks the answer by calculating its similarity to the model answer and returns feedback based on the similarity score.