### Lambda Function -> **dashQuestionEdit\_R**

API URL ->

https://j0mmgihtaj.execute-api.us-east-1.amazonaws.com/v1/dashQuestionEdit\_R

Resource -> **dashQuestionEdit\_R**

Stage -> v1

### 1. Role of the API

This API updates a quiz question (either MCQ or descriptive) in the database based on the provided questionId.

### 2. Functioning

* **Token Verification**: Validates the JWT token to authenticate the request.
* **CORS Handling**: Sets appropriate headers to handle Cross-Origin Resource Sharing (CORS).
* **Database Connection**: Ensures a consistent MongoDB connection.
* **Request Validation**: Ensures required fields like questionId and updatedQuestion are present and valid.
* **Question Update**: Updates the quiz question (either MCQ or descriptive) based on the provided questionId.
* **Response Handling**: Returns success or error messages based on the operation's outcome.

**3. Request Body**

**Headers:**

**Authorization**

**Value: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2NjcxYWU1NTZhNWY0YTRjNWNhMzMzYjUiLCJlbWFpbCI6InNhaW5pcHM5NDE0NjZAZ21haWwuY29tIiwiaWF0IjoxNzE5ODI1MDE3LCJleHAiOjE3MTk4NDMwMTd9.MDZDZZJPbJoNb9cBfb\_HRW\_7XPNDrituNsk-vwrJ3Ak**

**Body:**

**{**

**"questionId": "6681256ce68acf528d76aa6e",**

**"updatedQuestion": {**

**"question": "What is the capital of rajasthan?",**

**"options": [**

**{**

**"answer": "jaipur",**

**"answerImageLink": "https://example.com/berlin.jpg"**

**},**

**{**

**"answer": "delhi",**

**"answerImageLink": "https://example.com/madrid.jpg"**

**}**

**],**

**"correctAnswer": 2,**

**"description": "Select the correct capital city.",**

**"questionImageLink": "https://example.com/question-image.jpg"**

**}**

**}**

### 4. Response

**Success Response:**

* **Status Code**: 200

**{**

**"statusCode": 200,**

**"headers": {**

**"Access-Control-Allow-Origin": "\*",**

**"Access-Control-Allow-Headers": "Content-Type, Authorization",**

**"Access-Control-Allow-Methods": "OPTIONS,POST,GET"**

**},**

**"body": "{\"message\":\"MCQ question updated successfully\",\"questionType\":\"MCQ\"}"**

**}**

**Error Responses:**

* **Status Code**: 401
  + **Body**: {"message": "Unauthorized"} | {"message": "Unauthorized: Invalid token"}
* **Status Code**: 400
  + **Body**: {"message": "Invalid payload structure"} | {"message": "Invalid fields for the question update"} | {"message": "Invalid payload structure"}
* **Status Code**: 404
  + **Body**: {"message": "Question not found"}
* **Status Code**: 500
  + **Body**: {"message": "Internal Server Error"}

### 5. Logic

1. **Token Verification**: Validates the JWT token to ensure the user is authorized to perform the operation.
2. **CORS Headers**: Sets headers to allow CORS requests from any origin.
3. **Database Connection**: Ensures a consistent connection to MongoDB using Mongoose.
4. **Request Validation**: Ensures required fields like questionId and updatedQuestion are present and valid.
5. **Question Update**: Updates the quiz question (either MCQ or descriptive) based on the provided questionId.
6. **Response Creation**: Generates and returns appropriate responses based on the success or failure of the operation.

### 6. Dependencies

* **mongoose**: MongoDB object modeling tool.
* **jsonwebtoken**: Library for generating and verifying JSON Web Tokens (JWT).

CODE:

const mongoose = require('mongoose');

const jwt = require('jsonwebtoken');

const Quiz = require('./question.js'); // Adjust the path as needed

const DB\_URL = process.env.MONGODB\_URI;

const JWT\_SECRET = process.env.JWT\_SECRET\_KEY;

mongoose.connect(DB\_URL);

exports.handler = async (event) => {

// Handle preflight request

if (event.httpMethod === 'OPTIONS') {

return {

statusCode: 204,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online",

"Access-Control-Allow-Headers": "Content-Type, Authorization",

"Access-Control-Allow-Methods": "OPTIONS,POST,GET"

},

body: ''

};

}

// Handle main request

const token = event.headers.Authorization || event.headers.authorization;

if (!token) {

return {

statusCode: 401,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Unauthorized' }),

};

}

let payload;

try {

payload = JSON.parse(event.body);

} catch (error) {

console.error('Error parsing JSON payload:', error);

return {

statusCode: 400,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Invalid JSON payload' }),

};

}

const { questionId, updatedQuestion } = payload;

let decoded;

try {

decoded = jwt.verify(token, JWT\_SECRET);

} catch (error) {

console.error('Error verifying token:', error);

return {

statusCode: 401,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ error: 'Unauthorized: Invalid token' }),

};

}

// Validate the structure of the payload

if (!questionId || !updatedQuestion || typeof updatedQuestion !== 'object') {

return {

statusCode: 400,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Invalid payload structure' }),

};

}

// Ensure either question or questionImageLink is present

const hasValidQuestion = updatedQuestion.question || updatedQuestion.questionImageLink;

// Determine if the payload is for an MCQ or a descriptive question

const isMCQ = (

hasValidQuestion &&

updatedQuestion.options &&

updatedQuestion.correctAnswer &&

updatedQuestion.description !== undefined &&

Array.isArray(updatedQuestion.options) &&

updatedQuestion.options.length >= 2

);

const isDescriptive = (

hasValidQuestion &&

updatedQuestion.answer &&

!updatedQuestion.options && // Ensure no MCQ fields are present

!updatedQuestion.correctAnswer &&

!updatedQuestion.description

);

if (!isMCQ && !isDescriptive) {

return {

statusCode: 400,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Invalid fields for the question update' }),

};

}

try {

// Attempt to update the MCQ question

let quiz = null;

let message = '';

if (isMCQ) {

// Assign \_id to each option if not already present

updatedQuestion.options.forEach(option => {

if (!option.\_id) {

option.\_id = new mongoose.Types.ObjectId();

}

});

// Convert 1-based index to 0-based and assign option ID to correctAnswer

const correctAnswerIndex = updatedQuestion.correctAnswer - 1;

updatedQuestion.correctAnswer = updatedQuestion.options[correctAnswerIndex].\_id;

quiz = await Quiz.findOneAndUpdate(

{ 'mcqQuizz.\_id': questionId },

{

$set: {

'mcqQuizz.$.question': updatedQuestion.question,

'mcqQuizz.$.questionImageLink': updatedQuestion.questionImageLink,

'mcqQuizz.$.options': updatedQuestion.options,

'mcqQuizz.$.correctAnswer': updatedQuestion.correctAnswer,

'mcqQuizz.$.description': updatedQuestion.description

},

$inc: { 'mcqQuizz.$.version': 1 }

},

{ new: true }

);

message = 'MCQ question updated successfully';

} else if (isDescriptive) {

quiz = await Quiz.findOneAndUpdate(

{ 'descriptiveQuizz.\_id': questionId },

{

$set: {

'descriptiveQuizz.$.question': updatedQuestion.question,

'descriptiveQuizz.$.questionImageLink': updatedQuestion.questionImageLink,

'descriptiveQuizz.$.answer': updatedQuestion.answer,

'descriptiveQuizz.$.answerImageLink': updatedQuestion.answerImageLink

},

$inc: { 'descriptiveQuizz.$.version': 1 }

},

{ new: true }

);

message = 'Descriptive question updated successfully';

}

if (!quiz) {

return {

statusCode: 404,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Question not found' }),

};

}

return {

statusCode: 200,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message, questionType: isMCQ ? 'MCQ' : 'Descriptive' }),

};

} catch (error) {

console.error(error);

return {

statusCode: 500,

headers: {

"Access-Control-Allow-Origin": "https://admin.exambuilder.online"

},

body: JSON.stringify({ message: 'Internal Server Error' }),

};

}

};