### LAMBDA FUNCTION NAME -> **quizzDiscriptiveQuestionService**

### API URL-> <https://j0mmgihtaj.execute-api.us-east-1.amazonaws.com/v1/quizzDiscriptiveQuestionService>

Original URL -> <https://ee4pmf8ys1.execute-api.us-east-1.amazonaws.com/questionstyle/descriptiveQuestion>

API Name:Module2 Resource Name: descriptiveQuestion Stage name: v1

### 1. Role of the API

This API is designed to update descriptive quiz questions within an existing quiz. It verifies the user's authorization using a JWT token, validates the request payload, merges new descriptive quiz questions into an existing quiz document in MongoDB, and returns appropriate success or error responses.

**PAYLOAD ACCEPT** ->

**headers:**

"Authorization": eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2NjcxYWU1NTZhNWY0YTRjNWNhMzMzYjUiLCJlbWFpbCI6InNhaW5pcHM5NDE0NjZAZ21haWwuY29tIiwiaWF0IjoxNzE5NTcxOTI1LCJleHAiOjE3MTk1ODk5MjV9.yC0KNA7n\_atFpvHjuXvzxBKfB1cfRWsCxRpNxt6AWLE",

**body:**

{

"quizTitle": "Science10",

"descriptiveQuizz": [

{

"question": "Describe the process of photosynthesis.",

"questionImageLink": "image-url",

"answer": "Photosynthesis is a process used by plants to convert light energy into chemical energy.",

"answerImageLink": "image-url"

}

]

}

**PAYLOAD NOTE ->**

**1 -> question or questionImage are optional, but must pass atleast one**

**2-> answer or answerImage are optional, but must pass atleast one**

**3-> questionImageLink & answerImageLink contain url of image**

**RESPONSE PROVIDE** -> {

"statusCode": 201,

"headers": {

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

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

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

},

"body": "{\"message\":\"Descriptive question Saved Successfully\"}"

}

### 2. Functioning

* **Authorization**: Validates the JWT token passed in the Authorization header.
* **Validation**: Checks if the required fields (quizTitle and descriptiveQuizz) are present in the request payload and if descriptiveQuizz is an array. Validates each descriptive question object to ensure it contains either a question or a question image link, and either an answer or an answer image link.
* **Database Interaction**: Connects to MongoDB to find the quiz based on quizTitle and creatorEmail, merges the new descriptive quiz questions into the existing quiz, and saves the updated quiz document.
* **Response**: Returns appropriate HTTP status codes and JSON responses indicating success or failure.

### 5. Dependency

* **External Dependencies**: Uses MongoDB for data storage and jsonwebtoken for token verification.

### 6. Logic

* **Token Verification**: Verifies the JWT token to ensure the user is authorized.
* **Payload Validation**: Validates the structure and content of the incoming request payload.
* **Database Operations**: Finds the existing quiz in MongoDB based on quizTitle and creatorEmail, merges new descriptive questions into the quiz document, and saves it.
* **Error Handling**: Catches and handles specific errors such as missing token, invalid token, missing fields in payload, and database errors.

CODE:

//updated code

const mongoose = require('mongoose');

const Quiz = require('./question.js');

const jwt = require('jsonwebtoken');

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

let cachedDb = null;

const connectToDatabase = async () => {

if (cachedDb) {

return cachedDb;

}

try {

const connection = await mongoose.connect(process.env.MONGODB\_URI);

cachedDb = connection.connection.db;

return cachedDb;

} catch (error) {

throw new Error('Error connecting to MongoDB: ' + error.message);

}

};

exports.handler = async (event, context) => {

context.callbackWaitsForEmptyEventLoop = false;

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

return {

statusCode: 204,

headers: {

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

'Access-Control-Allow-Methods': 'OPTIONS, POST',

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

},

};

}

try {

await connectToDatabase();

const token = event.headers.Authorization;

if (!token) {

return createErrorResponse(401, 'Unauthorized: No token provided');

}

let decoded;

try {

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

} catch (error) {

return createErrorResponse(401, 'Unauthorized: Token expired or invalid');

}

if (!decoded.email) {

return createErrorResponse(400, 'Token does not contain an email');

}

const creatorEmail = decoded.email.toLowerCase();

const requestBody = JSON.parse(event.body);

const { quizTitle, descriptiveQuizz } = requestBody;

if (!quizTitle || !descriptiveQuizz || !Array.isArray(descriptiveQuizz)) {

return createErrorResponse(400, 'quizTitle and descriptiveQuizz are required fields and descriptiveQuizz should be an array');

}

// Validate Descriptive Question objects

for (let dq of descriptiveQuizz) {

if (!dq.question && !dq.questionImageLink) {

return createErrorResponse(400, 'Each Descriptive Question object must have either question or questionImageLink defined');

}

if (!dq.answer && !dq.answerImageLink) {

return createErrorResponse(400, 'Each Descriptive Question object must have either answer or answerImageLink defined');

}

}

let existingQuiz = await Quiz.findOne({ quizTitle, creatorEmail });

if (!existingQuiz) {

return createErrorResponse(404, 'Quiz with the provided title does not exist. Please provide a valid quiz title.');

}

// Merge descriptiveQuizz objects into existing quiz

existingQuiz.descriptiveQuizz = existingQuiz.descriptiveQuizz.concat(descriptiveQuizz);

await existingQuiz.save();

return createSuccessResponse(201, 'Descriptive questions saved successfully');

} catch (error) {

if (error.name === 'ValidationError') {

return createErrorResponse(400, 'Bad Request: ' + error.message);

}

return createErrorResponse(500, 'Internal Server Error: ' + error.message);

}

};

function createErrorResponse(statusCode, errorMessage) {

return {

statusCode,

headers: {

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

},

body: JSON.stringify({ error: errorMessage })

};

}

function createSuccessResponse(statusCode, message) {

return {

statusCode,

headers: {

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

},

body: JSON.stringify({ message })

};

}