### LAMBDA FUNCTION -> **paperSubmitCheck\_R**

API URL -> https://j0mmgihtaj.execute-api.us-east-1.amazonaws.com/v1/paperSubmitCheck\_R

Resource -> **paperSubmitCheck\_R**

Stage -> v1

### 1. Role of the API

This API is designed to handle the submission of a quiz paper, marking it as completed if either the MCQ or descriptive sections are fully completed.

### 2. Functioning

* **Token Verification**: Validates the JWT token provided in the request header to ensure authentication.
* **CORS Handling**: Sets appropriate headers to handle Cross-Origin Resource Sharing (CORS).
* **Database Connection**: Connects to MongoDB using Mongoose to interact with the quiz data.
* **Quiz Validation**: Checks if the specified quiz exists and if it is already marked as completed.
* **Completion Check**: Verifies if either the MCQ or descriptive quiz sections are fully completed.
* **Data Storage**: Marks the quiz as completed and saves the updated document in MongoDB.
* **Response Handling**: Returns appropriate success or error responses based on the outcome of the operations.

**3. Request Body**

**1. Headers:**

* **Authorization: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiI2NjcxYWU1NTZhNWY0YTRjNWNhMzMzYjUiLCJlbWFpbCI6InNhaW5pcHM5NDE0NjZAZ21haWwuY29tIiwiaWF0IjoxNzIxMTgwOTE2LCJleHAiOjE3MjExOTg5MTZ9.zhtMlw5sHYBNkG8NQCANzPZ6tg-I709KpEvSvF93dyA**
* **Content-Type: application/json**

**2. Body:**

**In Postman, you should use the raw option for the body and select JSON as the format. The body should be:**

**{**

**"quizTitle": "Jhhhjjh"**

**}**

### 4. Response

**Success Response:**

* **Status Code**: 200 {

Paper Submit Successfully

}

**Already Submit: 400**

**{**

**"statusCode": 400,**

**"headers": {**

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

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

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

**},**

**"body": "{\"message\":\"Paper is already submitted\"}"**

**}**

**Error Responses:**

* **Status Code**: 401
  + **Body**: {"error": "Unauthorized: token expired"}
* **Status Code**: 404
  + **Body**: {"message": "Quiz not found"}
* **Status Code**: 400
  + **Body**: {"message": "Paper is already submitted"}
  + **Body**: {"message": "Paper is not completed"}
* **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**: Connects to MongoDB using Mongoose for data interaction.
4. **Quiz Validation**: Checks if the quiz exists in the database and if it is already marked as completed.
5. **Completion Check**: Verifies if either the MCQ or descriptive quiz sections are fully completed based on the presence of questions.
6. **Data Storage**: Marks the quiz as completed if the completion check passes and saves it back to MongoDB.
7. **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'); // Assuming Quiz model is in models directory

const MONGO\_URI = process.env.MONGODB\_URI; // Set your MongoDB URI in environment variables

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

exports.handler = async (event) => {

// CORS headers for preflight requests

const preflightHeaders = {

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

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

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

};

// Function to handle preflight requests

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

return {

statusCode: 200,

headers: preflightHeaders,

body: JSON.stringify({ message: 'Preflight request handled' }),

};

}

// Validate JSON parsing and handle malformed payloads

let parsedBody;

try {

parsedBody = JSON.parse(event.body);

} catch (error) {

return {

statusCode: 400,

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

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

};

}

const { quizTitle } = parsedBody;

// Validate input: quizTitle should be a non-empty string

if (typeof quizTitle !== 'string' || quizTitle.trim() === '') {

return {

statusCode: 400,

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

body: JSON.stringify({ message: 'quizTitle is required and must be a non-empty string' }),

};

}

const token = event.headers.Authorization;

let decoded;

try {

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

} 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: token expired' }),

};

}

const creatorEmail = decoded.email.toLowerCase();

try {

await mongoose.connect(MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true });

// Find the quiz

const quiz = await Quiz.findOne({ quizTitle, creatorEmail });

if (!quiz) {

return {

statusCode: 404,

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

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

};

}

// Check if the paper is already submitted

if (quiz.isCompleted) {

return {

statusCode: 400,

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

body: JSON.stringify({ message: 'Paper is already submitted' }),

};

}

// Check if either mcqQuizz or descriptiveQuizz are fully completed

const isMcqCompleted = quiz.mcqQuizz && quiz.mcqQuizz.length > 0;

const isDescriptiveCompleted = quiz.descriptiveQuizz && quiz.descriptiveQuizz.length > 0;

if (isMcqCompleted || isDescriptiveCompleted) {

quiz.isCompleted = true;

await quiz.save();

return {

statusCode: 200,

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

body: JSON.stringify({ message: 'Paper submitted successfully' }),

};

} else {

return {

statusCode: 400,

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

body: JSON.stringify({ message: 'Paper is not completed' }),

};

}

} catch (error) {

console.error(error);

return {

statusCode: 500,

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

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

};

} finally {

await mongoose.disconnect();

}

};