Q & A WEBSITE USING REST API

DOCUMENTATION

BY

JAYANTH KUMAR

GitHub ID - SJ-Kumar

Code Overview

In this project, I created a backend Rest Api that includes basic functionalities that a simple Q&A website requires. This Api can be consumed by any frontend technologies such as React, Angular etc.

Dependencies:

- **expressjs:** The server for handling and routing HTTP requests
- **jsonwebtoken:** For generating JWTs used by authentication
- **mongoose:** For modeling and mapping MongoDB data to JavaScript
- **slugify:** For encoding titles into a URL-friendly format
- **bcryptjs:** For hashing passwords
- **dotenv**: Zero-Dependency module that loads environment variables
- **multer:** Node.js middleware for uploading files
- **nodemailer:** Send e-mails from Node.js

Application Structure:

- **server.js:** The entry point to our application. This file defines our express server and connects it to MongoDB using mongoose. It also includes the routes we'll be using in the application.
- **config/:** This folder contains configuration for central location environment variables and other configurations.
- **routes/:** This folder contains the route definitions (answer, question, etc.) for our API.
- **models/:** This folder contains the schema definitions for our Mongoose models (User, Question).
- **controllers/:** This folder contains controllers for our API.
- **public/:** This folder contains static files for our API.

- **middlewares/:** This folder contains middlewares for our API.
- **helpers/:** This folder contains helper functions for adapting 3rd party libraries for our API.
- **dummy/:** This folder contains dummy data created by dummy-generator.js file for our database.

Error Handling

In middlewares/errors/errorHandler.js, we define an error-handling middleware for handling Mongoose's errors and our own errors.

Authentication:

Requests are authenticated using the Authorization header and value "Bearer: {{token}}" with a valid JWT. We define express middlewares in middlewares/authorization/auth.js that can be used to authenticate requests. The required middlewares return 401 or 403 HTTP status codes if the authentication fails.

Functionalities of my REST API

Questions

Public Operations

- List all questions
  * Paginate and Limit number of Questions
  * Sorting Questions By Most-Answered, Most-Liked or More Recent(Default)
  * Searching Questions By Title
  * Population User Of The Question
- Get a single question with their answers

Private Operations

- Ask (Create) a New Question
  * Authenticated users only (Logged In Users)
  * Field validation
- Edit a Question
  * Owner User Only
  * Field Validation
- Delete a Question
  * Owner User Only
- Like a Question
  * Authenticated user only
  * Only 1 Like Per User
- Undo Like a Question
  * Authenticated user only
  * Only Applicable To Question That Liked Before

Answers

Public Operations

- Get All Answers by Question Id
- Get Single Answer By Answer Id

Private Operations

- Add (Create) a New Answer To Question
  * Authenticated users only (Logged In Users)
  * Field validation
- Edit a Answer
  * Owner User Only
  * Field Validation
- Delete a Answer
  * Owner User Only
- Like a Answer
  * Authenticated user only
  * Only 1 Like Per User
- Undo Like a Answer
  * Authenticated user only
  * Only Applicable To Answer That Liked Before

<u>**Users**</u>

Public Operations

- List all Users
  * Paginate and Limit number of Users
  * Search By name
- Get User Profile

Private Operations

- Block A User
- Delete A User

<u>**Authentication**</u>

- Authentication Strategy : JWT and Cookie
  * JWT and Cookie Expiration : 30 Minutes For Testing Api
- Registration
  * User can register as a "Admin" or simply "User"
  * Password Hash
  * Token includes : "id" and "name"
  * Token Are Stored In Cookie
- Login
  * User can login with "email" and "password"

  * Everytime a user login, new Token are sent to to client and stored in cookie.
- Logout
  * Token set to null in cookie.
- Forgot Password
  * Reset Password Token send to client via email.
  * This token expires in 1 hour.
- Reset Password
  * Reset Password Token can be used in 1 hour.
  * User can set a new password using this token.
- Update User Details (Bio)
  * Users can add their bio details when logged in.
- User Profile
  * Users can view their personal information after they login.
- Profile Photo Upload
  * Users can upload an avatar for their profile.

<u>Models</u>

User
- name
  * type : String
  * required : true
  * Validation : Please provide a name
- email
  * type : String
  * required : true
  * unique : true
  * Validation with Regex : Please provide a valid email
- role
  * type : String
  * enum : user,admin
  * default : user
- password
  * type : String
  * required : true
  * minlength : 6
  * Validation : Please provide a password
- createdAt
  * type : String
  * default : Date.now

- title
  * type : String
- about
  * type : String
- website
  * type : String
- place
  * type : String
- profile_image
  * type : String
- blocked
  * type : Boolean
  * default : false
- resetPasswordToken
  * type : String
- resetPasswordExpire
  * type : Date

<u>Question</u>

- title
  * type : String
  * required : true
  * Validation : Please provide a title
  * minLength : 10
  * unique : true
- content
  * type : String
  * required : true
  * Validation : Please provide a content
  * minLength : 20
- slug
  * type : String
- createdAt
  * type : Date
  * default : Date.now
- likeCount
  * type : Number
  * default : 0
  * min : 0

- likes
  * type : Array(ObjectId)
  * ref : "User"
- user
  * type : ObjectId
  * ref : "User"
- answerCount
  * type : Number
  * default : 0
- answers
  * type : Array(ObjectId)
  * ref : Answer

Answer

- content
  * type : String
  * required : true
  * Validation : Please provide a content
  * minLength : 20
- createdAt
  * type : Date
  * default : Date.now
- likeCount
  * type : Number
  * default : 0
  * min : 0
- likes
  * type : Array(ObjectId)
  * ref : User
- user
  * type : ObjectId
  * ref : User
  * required : true
- question
  * type : ObjectId
  * ref : Question
  * required : true

<u>**Middlewares**</u>

Authorization

- Middlewares That Protect Routes From Unauthorized Access
  * getAccessToRoute
  * getAdminAccess
  * getQuestionOwnerAccess
  * getAnswerOwnerAccess

Database

- Middlewares That Check Entities Exist With Given Ids
  * checkQuestionAndAnswerExist
  * checkQuestionExist
  * checkUserExist

Error

- Middleware That Captures All Errors
  * errorHandler

Query

- Middleware That Provides Advance Query Functionalities
  * answerQueryMiddleware
  * questionQueryMiddleware
  * userQueryMiddleware

Security

- Middleware That Provides Security to Rest Api
  * limitAccess
  * hpp
  * cors
  * helmet
  * mongoSanitize

<u>Helper Functions and Classes</u>

Database

- connectDatabase
  * MongoDb Connection

Error

- customError
  * Customized Error Class
- errorWrapper
  * Function that catches asynchronous errors

3rd Party Libraries

- photoUpload
  * Helper Function That Customized Upload Process with Multer Package
- sendEmail
  * Helper Function That Customized
  Mail Process with NodeMailer Package

<u>Environment Variables and Constants</u>

Environment Variables and Constants Can Be Set in ./config/env/config.env.