Relevel RTO Exam Question App

Problem Statement -

Hello folks, Relevel, a software engineering firm is looking to design a rto exams question app. You have to design & build the backend for the rto exams question app with the ability for the users to register, login, add question answers and the symbols for study of the online exam of RTO. Your task is to go through the Problem Statement and the Requirements and create a rto exams question app for the Relevel.

The tech stack required is **Nodejs, Express JS and MongoDB**. Preferred IDE is **Visual Studio Code** to import the project directly. **Postman** tool to test the REST APIs. Make sure **npm and git** etc are configured on local.

How to Set up the Application:

To start the development of template repo is provided at the Github link
https://github.com/abhishek36/rto-exam-question-app-nodejs-development

Steps to follow:

- Install Git and Nodejs on Local system if not already installed and set env variable for both.
- Fork the above repo and clone on local system(`git clone <repo>`)
- Create .env file and store all the environment variables inside this file only
- Installed all the necessary dependencies by running `npm install`.
- Finally, run `npm run dev` to start the development server .

Start coding now and to check the hosted app in the browser, they can search for http://localhost:8000/ in their browser. This ensures the setup is complete.

Template 1 (Authentication)

1. Story 1 - (15 mins)

The task is to create a User model to store key information of all the users. The various functionalities to develop under this story are:

- The user model should be able to store Name, Email/Phone, Password, Avatar , Username, Role, Gender & Date of Birth.
- Username and Email/Phone should be unique

2. Story 2 - (30 mins)

The task is to create an API endpoint "**localhost:8000/authentication/register**" for the registration of the user:

- The API request should take the input of Name, Email/Phone, Password, Avatar, Username, Role, Gender & Date of Birth.
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- Display status & response code in the response.
- The successful request should return a 201 response code along with the message "User is registered successfully".
- Create proper validation and give proper message i.e "Email is incorrect" , "Password is incorrect"

3. Story 3- (30 mins)

The task is to create an endpoint **localhost:8000/authentication/login** so that we can verify if user credentials are valid .

- The API request should take the input of Email or Username & Password.
- Authenticate the value against the data stored in the database and upon successful authentication return appropriate HTTP status with JWT token.
- While creating JWT store the email, username and userId of the userId
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- Display status & response code in the response.
- In case the user has provided a wrong password , return a HTTP status as 401 with a message as "Invalid Password".
- If the user name provided by the user doesn't exist return a HTTP status as 404 with message "User does not exist"
- The successful request should return 200 response codes along with the JWT token which will be used for authorization in subsequent requests.

4 . Story 4- (20 mins)

The task is to create an endpoint **localhost:8000/authentication/getProfile** so we can get all the information of the user .

- Authenticate the value against the data stored in the database and upon successful authentication return appropriate HTTP status with JWT token.
- Get the userId from the JWT
- Use an appropriate HTTPS
- method out of GET, POST, PUT, DELETE, etc.
- The successful request should return 200 response codes along with the JWT token which will be used for authorization in subsequent requests.

5 . Story 5- (20 mins)

The task is to create an endpoint **localhost:8000/authentication/updateProfile** so we can get all the information of the user .

- Authenticate the value against the data stored in the database and upon successful authentication return appropriate HTTP status with JWT token.
- Get the userId from the JWT for updating the profile
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- The successful request should return 200 response codes along with the JWT token which will be used for authorization in subsequent requests.
- Display status & response code in the response.

6. Story 6 - (20 mins)

The task to create auth middleware. We have to add security as well as we have to maintain session throughout the login. Store Role for user model in JWT so we will add restrictions on the basis of role.

- Create auth middleware in auth.middleware.js file.
- We have to check if the token is provided or not whenever the request is for any API.
- Give the proper response with the response code and message
- And if the token is provided we have to verify the token is valid or not using the secret key that is in the .env file.

Template 2 (Question)

1. Story 1 - (15 mins)

The task is to create the required models using the name “Question”:

- The model should be able to store the question and the answers.
- Appropriate data types and constraints have to be used and always use timestamps..
- Exports the question model properly so we can use it further.

2. Story 2 - (30 mins)

The task is to create an API endpoint “**localhost:8000/question/add**” so only admin role can add the question and answer:

- Use the auth middleware in between the API and check for the role only admin can add.
- There should be a key-value pair of “Authorization: <JWT_TOKEN>” & if it's incorrect or not present, the API should throw a Bad Request Exception with 401 Unauthorized code.
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- Display status & response code in the response.

3. Story 3 - (20 mins)

The task is to create an API endpoint “**localhost:8000/question/getAll**” to list the question:

- The API request will give the array of questions.
- Add proper pagination logic.
- There should be a key-value pair of “Authorization: <JWT_TOKEN>” & if it’s incorrect or not present, API should throw Bad Request Exception with 401 Unauthorized code.
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- Display status & response code in the response.

4. Story 4 - (10 mins)

The task is to create an API endpoint “**localhost:8000/question/:id**” to get, update and delete record

- This API will accept Id in the API url. And based on the method we will differentiate the API is for get , update or delete
- Only the admin role can update and delete records.
- Set GET method for get the record, PUT method for the update and DELETE method for delete the record
- Display the proper response with status code.

Template 3 (Symbol)

1. Story 1 - (15 min)

The task is to create the required models “Symbol ”.

- The model should be able to store the various types of symbols with their title and description.
- The model will store title, ImageURL and description.
- Appropriate data types and constraints have to be used.
- Use Timestamps for sorting the records.

2. Story 2 - (20 mins)

The task is to create an API endpoint “**localhost:8000/symbol/add**” only admin can add the symbol:

- There should be a key value pair of “Authorization: <JWT_TOKEN>” & if its incorrect or not present API should throw Bad Request Exception with 401 Unauthorized code.
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- The successful request should return 201 response code.

3. Story 3 - (20 mins)

The task is to create an API endpoint “**localhost:8000/symbol/getAll**” to list the symbol:

- The API request will give the array of symbols with title and description.
- There should be a key-value pair of “Authorization: <JWT_TOKEN>” & if it’s incorrect or not present, API should throw Bad Request Exception with 401 Unauthorized code.
- Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
- Display status & response code in the response.

4. Story 4- (20 mins)

The task is to create an API endpoint “**localhost:8000/symbol/:id**” to update and delete record

- This API will accept Id in the API url. And based on the method we will differentiate the API is for get , update or delete
- Only the admin role can update and delete records.
- Set GET method for get the record, PUT method for the update and DELETE method for delete the record
- Display the proper response with status code.