Relevel In-Shorts Clone

Problem Statement -

Hello folks, Relevel, a software engineering firm is looking to design a news app. You have to design & build the backend for the news application with the ability for the users to register, login,view the news in categories and bookmark the news.

Your task is to go through the Problem Statement and the Requirements and create an in-shorts clone 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 a template repo is provided at the Github link – https://github.com/abhishek36/relevel-in-shorts-backend-nodejs

Steps to follow:

1. Install Git and Nodejs on Local system if not already installed and set env variable for both.
2. Fork the above repo and clone on local system(`git clone <repo>`)
3. Create .env file and store all the environment variables inside this file only
4. Installed all the necessary dependencies by running `npm install`.
5. 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

1. Story 1 - (30 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:

a. The user model should be able to store Name, Email/Phone, Password, Username, Gender & Date of Birth.
b. Username and Email/Phone should be unique
c. Appropriate constraints including the primary key should be there.

2. Story 2 - (30 - 40 mins)

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

a. The API request should take the input of Name, Email/Phone, Password, Username, Role, Gender & Date of Birth.
b. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
c. Display status & response code in the response.
d. The successful request should return a 201 response code along with the message “User is registered successfully”.
e. 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.
- 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.

Template 2

1. Story 1 - (30 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.

a. Create auth middleware in auth.middleware.js file.

b. We have to check if the token is provided or not whenever the request is for any API.
c. Give the proper response with the response code and message
d. 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 3

Story 1 - (30 mins)

The task is to create the required models using the name “Category” to store the categories of news. The various functionalities to develop under this story are:

a. The model should be able to store the category with column title, imageURL and description.
b. Appropriate data types and constraints have to be used and always use timestamps..
c. Only role admin can add the category.

2. Story 2 - (40-50 mins)

The task is to create an API endpoint “**localhost:8000/category/add**” so that admin can add the category:

a. Use the auth middleware in between the API and check for the role admin.
b. 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.
c. If the role is other than admin throw and response that only admin can create categories with proper status code.
d. The API request should take the input of the title and description in the request body.
e. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
f. Display status & response code in the response.

3. Story 3 - (20 mins)

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

a. The API request will give the array of categories with each object containing two columns title and description.
b. 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.
c. Apply proper validations for the value, for example, if the username is not there in the User table, it should throw a 404 Not Found Exception.
d. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.

e. Display status & response code in the response.

Template 4

1. Story 1 - (30 mins)

The task is to create the required models to store the “news” by the admin. The various functionalities to develop under this story are:

a. The model should be able to store the news added by the admin.
b. It should be normalised so as to make sure we don’t store redundant data.
c. Appropriate data types and constraints have to be used and the news have to be stored in relation with category.

2. Story 2 - (20 mins)

The task is to create an API endpoint “**localhost:8000/news/add**” to store a news by the admin:

a. The API request should take the input of the categoryId, title, imageURL and description body in the request payload.
b. 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.
c. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
d. Display status & response code in the response.

3. Story 3 - (20-30 mins)

The task is to create an API endpoint “**localhost:8000/news/getAll**” to get all news:

a. 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.
b. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
c. If there’s no news, just return an empty response.
d. The successful request should return 200 response codes along with the list of news of various categories.

4. Story 4- (30 mins)

The task is to create an API endpoint “**localhost:8000/news/get/:categoryId**” so we can get the news of particular category :

a. The API request should take the input of the categoryId in request param in the api url.

b. 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.
c. Apply proper validations for the value, for example, if the username is not there in the User table, it should throw a 404 Not Found Exception.
d. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.\
e. Display status & response code in the response.
f. The successful request should return 201 response code

Template 5

1. Story 1 - (20-30 mins)

The task is to create the required models using the name “BookMark” to make the news bookmark by the user just like adding to cart functionality in e-commerce websites.

various functionalities to develop under this story are:

a. The model should be able to store the news in a bookmark table added by the user.
b. It should be normalised so as to make sure we don’t store redundant data.
c. Appropriate data types and constraints have to be used and the news have to be stored in relation to the user.

2. Story 2 - (20-30 mins)

The task is to create an API endpoint “**localhost:8000/bookmark/add**” to add news to the bookmark of a particular user:

a. The API request should take the input of the userId of a user and newsId of a news item that we want to add in the bookmark as a request body..
b. 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.
c. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
d. The successful request should return 201 response code.

3. Story 3 - (20-30 mins)

The task is to create an API endpoint “**localhost:8000/bookmark/:userId**” so we can get the bookmark new of the particular user:

a. The API request should take the input of the userIdof a user as a request param in the api url.

b. 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.
c. Use an appropriate HTTPS method out of GET, POST, PUT, DELETE, etc.
d. Display status & response code in the response.
e. The successful request should return a 200 response with the list ok bookmark item.

Template 6

You can get the idea of database from the below link
https://drive.google.com/file/d/1WRI9sph3q_A0d4iAkRcoeAtHiBwvXNTU/view