JuniorDevInterviewPrep

This REST API was created to help junior backend developers prepare for interviews. The application has two main study methods: flashcards and code challenges. The admin user is responsible for creating code challenge categories and adding relevent code challenges inside of the related code challenge category. The admin is also responsible for creating flashcard sets with relevant flashcards inside of it. Users with the role "member" have access to the admin's flashcard sets, but they cannot add new flashcards to the admin's flashcard sets. The member users can create their own flashcard sets/flashcards, but they cannot create code challenge categories/code challenges. The API uses JWT authentication, and to access the endpoints, authentication is required. Member users cannot view other member users data.

AUTHORIZATION Bearer Token

Token	<token>

codeChallenges

These endpoints require client authorization. An authorized (logged in) client can retrieve and practice all code challenges. A user with the 'admin' role can create, update, and delete them.

All endpoints require a bearer token in the header.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection JuniorDevInterviewPrep

GET Code Challenges

http://localhost:3001/codeChallenges

Description: Retrieves all code challenges from the database.

Response: The success code is 200 and an array of the code challenge objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

POST Code Challenge

http://localhost:3001/codeChallenges

Description: Creates a new code challenge.

Body parameters:

- `question` (string, required): The code challenge question or task.
- `solution` (string, required): The solution to solving the code challenge.
- `hint` (string, required): Tips for how to solve the challenge.
- `betterSolution` (string, optional): The optimal solution to the challenge.
- `CodeChallengeCategoryId` (string, required): ID of the associated category.
- `UserId` (string, required): ID of the user creating the code challenge.

Response: The success code is 200 and an array of the code challenge objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "question": "Create a for loop that prints number 0-10",
    "solution": "for(let i = 0; i <= 10; i++) {\n\tconsole.log(i);\n\t}",
    "hint": "use 'for'",
    "betterSolution": "Loops are simple and useful. Sometimes a while loop can be better, but
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db",
    "CodeChallengeCategoryId": "8beba98c-5a42-479c-b61a-40760c3a0b02"
}
```

GET Code Challenge

http://localhost:3001/codeChallenges/f93a0b5e-7433-4131-958b-9b9eb2835042

Description: Retrieves a specific code challenge from the database.

URL parameter: `id` (string, required): The id of the code challenge that is being requested.

Response: The success code is 200 and the code challenge object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

PUT Code Challenge

http://localhost:3001/codeChallenges/f93a0b5e-7433-4131-958b-9b9eb2835042

Description: Updates an exisiting code challenge.

URL parameter: `id` (string, required): The id of the code challenge that is being updated.

Body parameters:

- `question` (string, required): The code challenge question or task.
- `solution` (string, required): The solution to solving the code challenge.
- `hint` (string, required): Tips for how to solve the challenge.
- `betterSolution` (string, optional): The optimal solution to the challenge.
- `CodeChallengeCategoryId` (string, required): ID of the associated category.
- `UserId` (string, required): ID of the user creating the code challenge.

Response: The success code is 200 and an array of the code challenge objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "question": "Create a for loop that prints number 0-10",
    "solution": "for(let i = 0; i <= 10; i++) {\n\tconsole.log(i);\n\t}",
    "hint": "use 'for'",
    "betterSolution": "Loops are simple and useful. Sometimes a while loop can be better, b
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db",
    "CodeChallengeCategoryId": "8beba98c-5a42-479c-b61a-40760c3a0b02"
}
```

DELETE Code Challenge

http://localhost:3001/codeChallenges/f93a0b5e-7433-4131-958b-9b9eb28350

Description: Deletes a specific code challenge from the database.

URL parameter: `id` (string, required): The id of the code challenge that is being deleted.

Response: The success code is 200 and an array of the code challenge objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

codeChallengeCategories

These endpoints require client authorization. An authorized (logged in) client can retrieve and practice all code challenge categories. A user with the 'admin' role can create, update, and delete them.

All endpoints require a bearer token in the header.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection JuniorDevInterviewPrep

GET Code Challenge Categories

http://localhost:3001/codeChallengeCategories

Description: Retrieves all code challenge categories from the database.

Response: The success code is 200 and an array of the code challenge category objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

GET Code Challenge Category

http://localhost:3001/codeChallengeCategories/8beba98c-5a42-479c-b61a-40760c3a0b02

Description: Retrieves a specific code challenge category from the database.

URL parameter: `id` (string, required): The id of the code challenge category that is being requested.

Response: The success code is 200 and the code challenge category object is included, inlcuding the associated code challenges. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

POST Code Challenge Category

http://localhost:3001/codeChallengeCategories

Description: Creates a new code challenge category.

Body parameters:

- `name` (string, required): The code challenge category name.
- `UserId` (string, required): ID of the user creating the code challenge.

Response: The success code is 200 and the created code challenge category object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "name": "Programming Foundations",
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db"
}
```

PUT Code Challenge Cateogry

http://localhost:3001/codeChallengeCategories/8beba98c-5a42-479c-b61a-40760c3a0b02

Description: Updates an existing code challenge category.

Body parameters:

- `name` (string, required): The code challenge category name.
- `UserId` (string, required): ID of the user creating the code challenge.

Response: The success code is 200 and the updated code challenge category object is included. If the request fails an error message and error code are included.

error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

```
json

{
    "name": "Programming Foundations",
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db"
}
```

DELETE Code Challenge Category

http://localhost:3001/codeChallengeCategories/8beba98c-5a42-479c-b61a-40760c3a0b02

Description: Deletes a specific code challenge category and the associated code challenges from the database.

URL parameter: `id` (string, required): The id of the code challenge category that is being deleted.

Response: The success code is 200 and and a success message are included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

flashcards

These endpoints require client authorization. An authorized (logged in) client can retrieve and practice the admin's flashcards. The user can also create and practice using their own flashcards. A user with the 'admin' role can create, update, and delete the flashcards.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection JuniorDevInterviewPrep

GET Flashcards

http://localhost:3001/flashcards

Description: Retrieves all flashcards from the database.

Response: The success code is 200 and an array of the flashcard objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

GET Flashcard

http://localhost:3001/flashcards/5ed28e15-5786-4ad8-b5ac-5a8e04f2083f

Description: Retrieves a specific flashcard from the database.

URL parameter: `id` (string, required): The id of the flashcard that is being requested.

Response: The success code is 200 and the flashcard object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

POST Flashcard

http://localhost:3001/flashcards

Description: Creates a new flashcard.

Body parameters:

- `question` (string, required): The flashcard question.
- `answer` (string, required): The answer to the flashcard question.
- `FlashcardSetID` (string, required): ID of the associated flashcard set.
- `UserId` (string, required): ID of the user creating the flashcard.

Response: The success code is 200 and the flashcard object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

Body raw (json)

```
json

{
    "question": "What does API stand for?",
    "answer": "Application Programming Interface",
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db",
    "FlashcardSetId": "fd6662e1-7194-4d4d-a1c9-327c18424395"
}
```

PUT Flashcard

http://localhost:3001/flashcards/5ed28e15-5786-4ad8-b5ac-5a8e04f2083f

Description: Creates a new flashcard.

URL parameter: `id` (string, required): The id of the flashcard being updated.

Body parameters:

- `question` (string, required): The flashcard question.
- `answer` (string, required): The answer to the flashcard question.
- `FlashcardSetID` (string, required): ID of the associated flashcard set.
- `UserId` (string, required): ID of the user creating the flashcard.

Response: The success code is 200 and the updated flashcard object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

```
json

{
    "question": "What does API stand for?",
    "answer": "Application Programming Interface",
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db",
    "FlashcardSetId": "fd6662e1-7194-4d4d-a1c9-327c18424395"
}
```

DELETE Flashcard

http://localhost:3001/flashcards/5ed28e15-5786-4ad8-b5ac-5a8e04f2083f

Description: Retrieves a specific flashcard from the database.

URL parameter: `id` (string, required): The id of the flashcard that is being deleted.

Response: The success code is 200 and a success message is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

flashcardSets

These endpoints require client authorization. An authorized (logged in) client can retrieve and practice the admin's flashcard sets. The user can also create and practice using their own flashcard sets. A user with the 'admin' role can create, update, and delete the flashcard sets.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection JuniorDevInterviewPrep

GET Flashcard Sets

http://localhost:3001/flashcardSets/set?page=1&size=2

Description: Retrieves all flashcard sets from the database.

Query params:

- `Page` : The page number of the flashcard set.
- `Size` : The number of flashcard sets per page.

Response: The success code is 200 and an array of the flashcard set objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

PARAMS

page	1
size	2

GET Flashcard Set List

http://localhost:3001/flashcardSets/list/fd6662e1-7194-4d4d-a1c9-327c18424395

Description: Retrieves the specific flashcards in a particular flashcard set from the database.

URL parameter: `id` (string, required): The id of the flashcard set that is being requested.

Response: The success code is 200 and the flashcard set object with its associated flashcards are included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

POST Flashcard Set

http://localhost:3001/flashcardSets

Description: Creates a new flashcard set.

Body parameters:

- `name` (string, required): The name of the flashcard set.
- `UserId` (string, required): ID of the user creating the flashcard set.

Response: The success code is 200 and the flashcard set object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "name": "Admin Set",
```

```
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db"
}
```

PUT Flashcard Set

http://localhost:3001/flashcardSets/fd6662e1-7194-4d4d-a1c9-327c18424395

Description: Updates an existing flashcard set.

Body parameters:

- `name` (string, required): The name of the flashcard set.
- `UserId` (string, required): ID of the user creating the flashcard set.

Response: The success code is 200 and the updated flashcard set object is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "name": "admin Set",
    "UserId": "3dc83de4-20ab-476f-8b97-fb69289d16db"
}
```

DELETE Flashcard Set

http://localhost:3001/flashcardSets/fd6662e1-7194-4d4d-a1c9-327c18424395

Description: Deletes a specific flashcard set from the database.

URL parameter: `id` (string, required): The id of the flashcard set that is being deleted.

Response: The success code is 200 and a success message is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

users

Users can create an account using the /signup enpoint, then they can use /login to login. After successful login, the JWT token is returned in the JSON response. This token allows users to access all functionalities of the API. The user has a role of 'admin' or 'member'. Users can also reset their password by requesting an email with a link to reset their password.

AUTHORIZATION Bearer Token

This folder is using Bearer Token from collection JuniorDevInterviewPrep

POST Signup

http://localhost:3001/signup

Description: Creates a new user with the provided data.

Body parameters:

- name (string, required): The name of the new user.
- email (string, required): The unique email of the new user.
- password (string, required): The password of the new user.

Response: The status is "success" and the message is "Successfully signed up." If the status is "fail" an error message is included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "name": "Admin",
    "email": "brittany.bolling@aol.com",
    "password": "4321"
}
```

POST Login

http://localhost:3001/login

Description: Logs in a user with the provided data, and generates a new token.

Body parameters:

- email (string, required): The unique email of the user.
- password (string, required): The password of the user.

Response: The status is "success", result is "Successfully logged in", and the created user object. The user's unique JWT token is generated and included in the user object after logging in. If the status is "fail" an error message is included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "email": "admin@gmail.com",
    "password": "4321"
}
```

POST Logout

http://localhost:3001/logout

Description: When the user is finished studying they can choose to logout, and their JWT token will be deleted.

Response: The status code and message that matches if the request failed or succeeded.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "email" : "admin@gmail.com"
}
```

GET Users

http://localhost:3001/Users

Description: Retrieves all users from the database.

Response: The success code is 200 and an array of the user objects is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

POST Forgot Password

http://localhost:3001/forgotPassword

Description: Sends a reset link to the user's email.

Body parameters:

- email (string, required): The unique email of the user.

Response: The success code is 200 and an email is sent to the user with a link to reset their password. A unique reset token is sent to the user. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "email": "brittany.bolling@aol.com"
}
```

PUT Reset Password

http://localhost:3001/resetPassword/fa10cdb733ff0cd0ec14af63165ca64733d1e055

Description: Logs in a user with the provided data, and generates a new token.

Path Parameter: `resetToken` : Generated and sent to the user's email that is associated with their account.

Body parameters:

- newPassword (string, required): The new password of the user.
- confirmPassword (string, required): Confirmation of the new password of the user.

Response: The success code is 200 and the user's password is updated. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

Body raw (json)

json

```
{
    "newPassword": "newpass",
    "confirmPassword": "newpass"
}
```

GET Reset Link

http://localhost:3001/resetPassword/fa10cdb733ff0cd0ec14af63165ca64733d1

Description: Retrieves a new reset token from the database.

URL parameter: `resetToken` (string, required): The reset token newly generated for the user to reset their password.

Response: The success code is 200 and a mesage the users password was reset. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep

DELETE User

http://localhost:3001/3dc83de4-20ab-476f-8b97-fb69289d16db

Description: Deletes a specific user from the database.

URL parameter: `id` (string, required): The id of the user that is being deleted.

Response: The success code is 200 and a success message is included. If the request fails an error message and error code are included.

AUTHORIZATION Bearer Token

This request is using Bearer Token from collection JuniorDevInterviewPrep