Service Layers

Lizzie Maczka

November 5, 2023

Overview

For my capstone project, I will be using Express.js in conjunction with Heroku to serve as the back-end framework of my web application. The Express.js application will be deployed on Heroku and will act as a link between the front-end user interface and the PostgreSQL database.

Endpoints

Users

Register a new user (POST):

URL: https://characternotewook/api/users/register

Users will be required to register an account to use Character Notebook. Clients will provide a username, password, and email address, and the server will generate a user account with the provided information.

Sample request:

```
{
  "username": "johndoe",
  "password": "password123",
  "email": "johndoe@example.com"
}
```

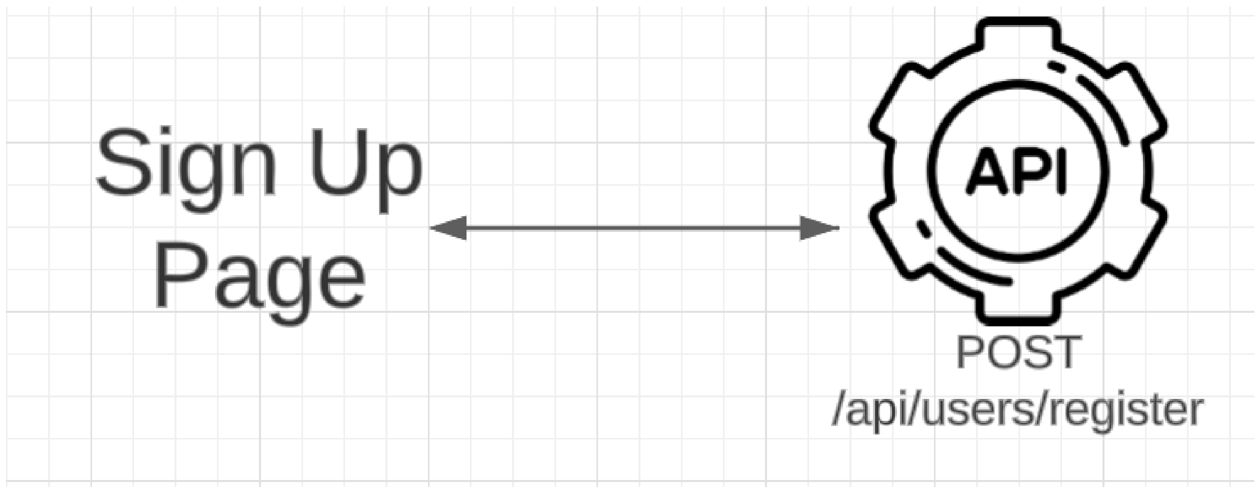
Successful response:

```
{
  "message": "User registered successfully",
  "userId": "123456"
}
```

Error response:

```
{
  "error": "Invalid username or password"
}
```

Diagram:



Log in a user (POST):

URL: https://characternotebook.com/api/users/login

This endpoint will handle login requests. Users will provide a username and password, and the server will validate the credentials provided. Upon successful validation, the user will be logged in to their account.

Sample request:

```
{
  "username": "johndoe",
  "password": "Abc123"
}
```

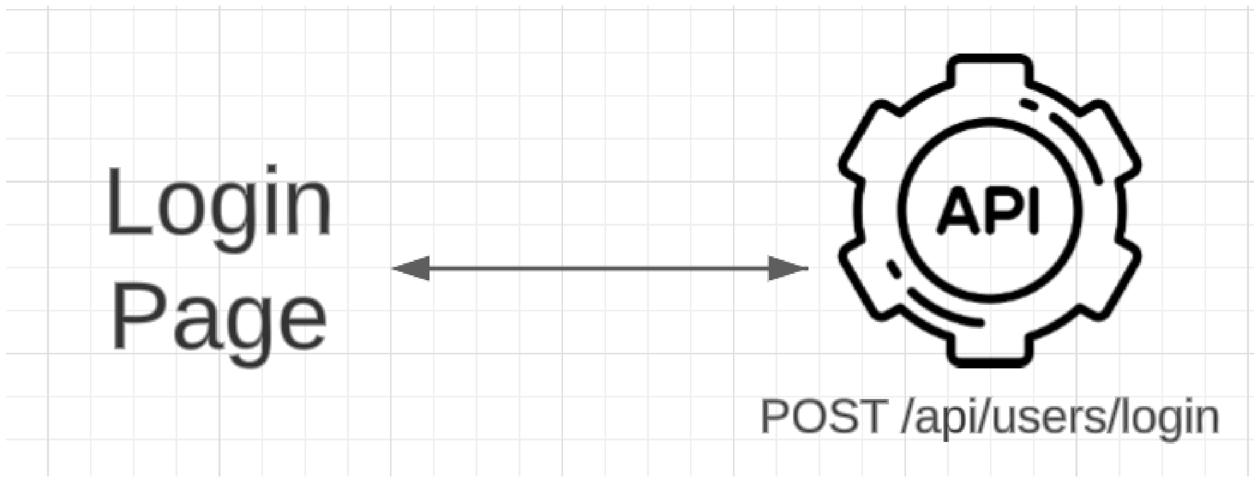
Successful response:

```
{
  "message": "Login successful",
  "user": {
    "id": 123,
    "username": "johndoe",
    "email": "johndoe@example.com"
  }
}
```

Error response:

```
{
  "message": "The username or password is incorrect. Please try again."
}
```

Diagram:



Log out the current user (POST):

URL: https://characternotebook.com/api/users/logout

When users are ready to end their session, this endpoint will log the user out. The user's session will be invalidated, and re-authentication will be required to log back in.

Sample request:

```
POST https://characternotebook.com/api/users/logout
```

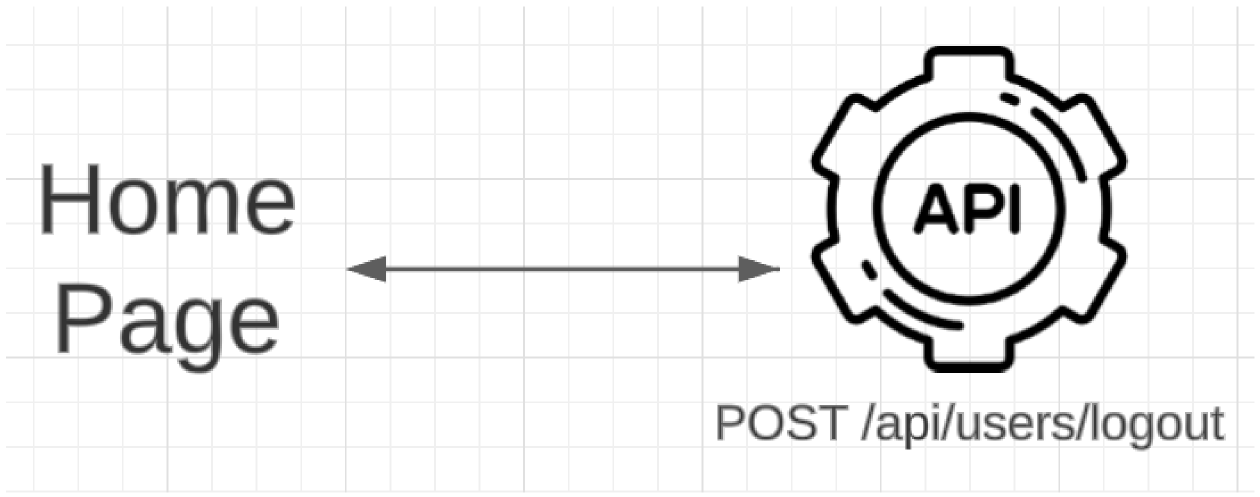
Successful response:

```
{
  "message": "Logout successful"
}
```

Error response:

```
{
  “message”: “Something went wrong. Please try again.”
}
```

Diagram:



Characters

Retrieve list of characters owned by the user (GET):

URL: https://characternotebook.com/api/characters

After a successful login, users will be able to view a list of all the characters they own. This endpoint will be used to fetch a list of all their characters.

Sample request:

GET https://characternotebook.com/api/characters

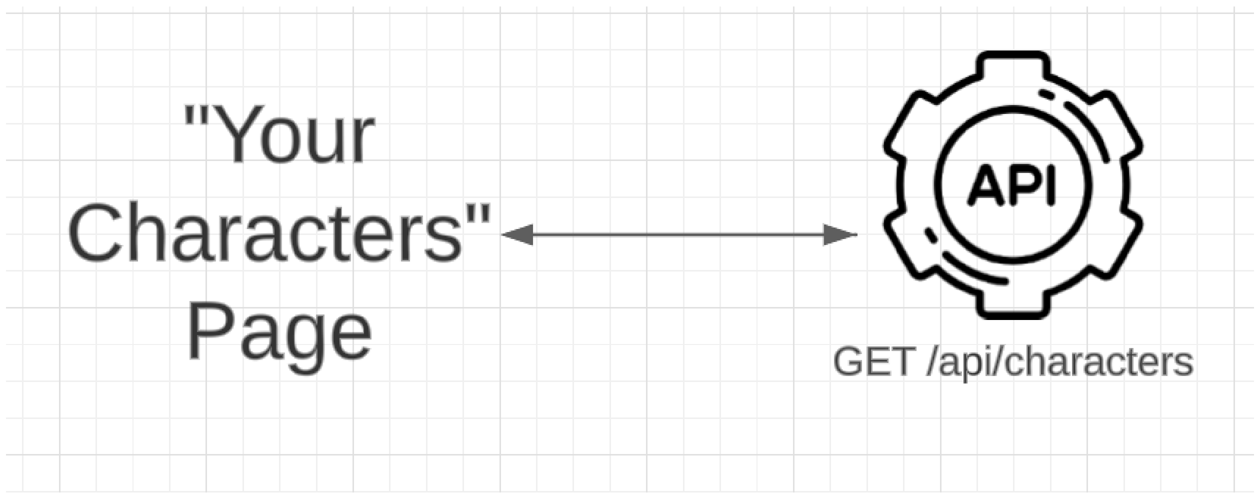
Success response:

```
{
  "characters": [
    {
      "name": "Jane Johnson"
    },
   {
      "name": "Mike Michaelson"
    },
 {
      "name": "Eliza Elsebeth"
    },
  ]
 }
```

Error response:

```
 {
    "message": "Something went wrong. Please try again."
 }
```

Diagram:



Create a new character (POST):

URL: https://characternotebook.com/api/characters/newcharacter

One of the main features of Character Notebook is the ability for users to add new characters to their profile. This endpoint will be used to submit data when creating a new character, such as the character’s various attributes.

Sample request:

```
{
  "name": "Jane Johnson",
  "age": 25,
  "gender": "Female",
  "species”: “human",
  “eyes”: “blue”,
 “hair”: “brown and long”
}
```

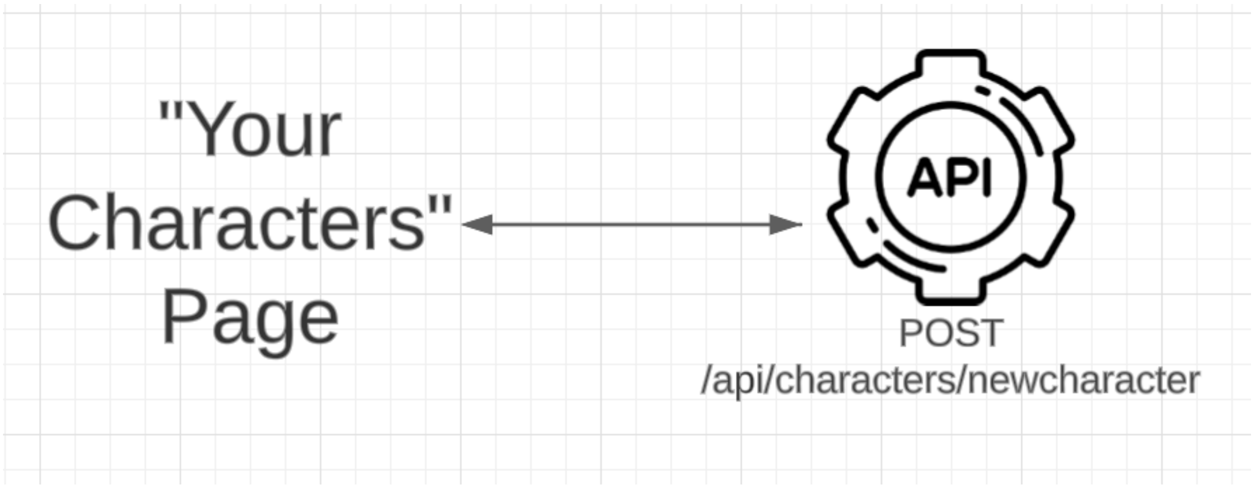
Success response:

```
{
  "message": "Character created.",
  "characterId": 654321
}
```

Error response:

```
{
  "error": "Please input all required information."
}
```

Diagram:



Retrieve details of a specific character (GET):

URL: https://characternotebook.com/api/characters/:name

After creating a character, users will be able to view the information they provided about that character. This is done by navigating to the specific character's Character Profile. This endpoint will be used to retrieve and display the information the user provided about the character.

Sample request:

```
GET https://characternotebook.com/api/characters/:name
```

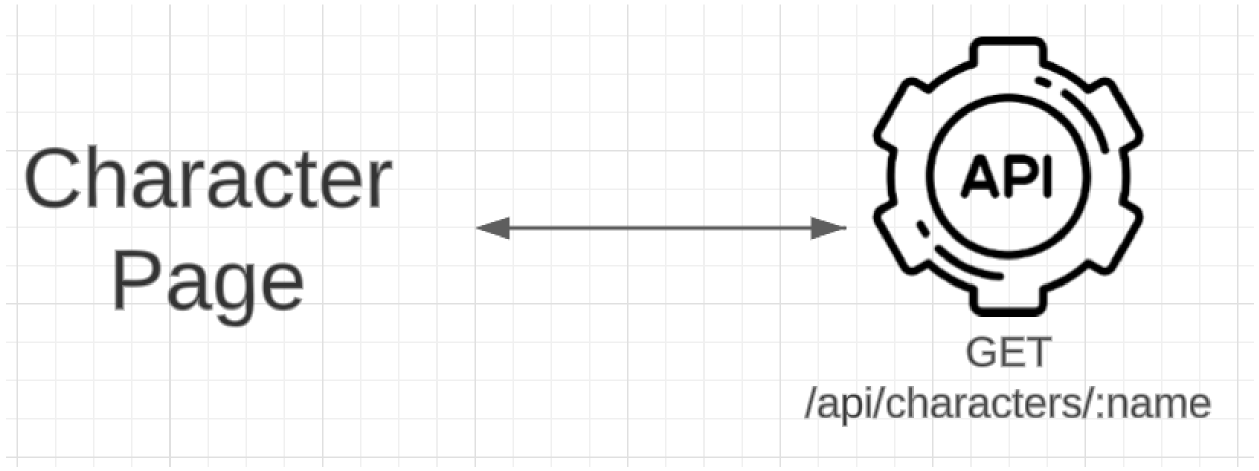
Success response:

```
{
  "name": "Jane Johnson",
  "age": 25,
  "gender": "Female",
  "species": “human",
  “eyes”: “blue”,
 “hair”: “brown and long”
}
```

Error response:

```
{
  "message": "Character does not exist."
}
```

Diagram:



Search for a specific character (GET):

URL: https://characternotebook.com/api/characters/search/:name

Users will be able to view a list of all of their existing characters. However, depending on how many characters they have created, it can be challenging to find one specific one. Because of this, Character Notebook will contain a search bar for users to use to pull up a specific character from the list. This endpoint will be responsible for handling search queries.

Sample request:

```
GET https://characternotebook.com/api/characters/search/:name
```

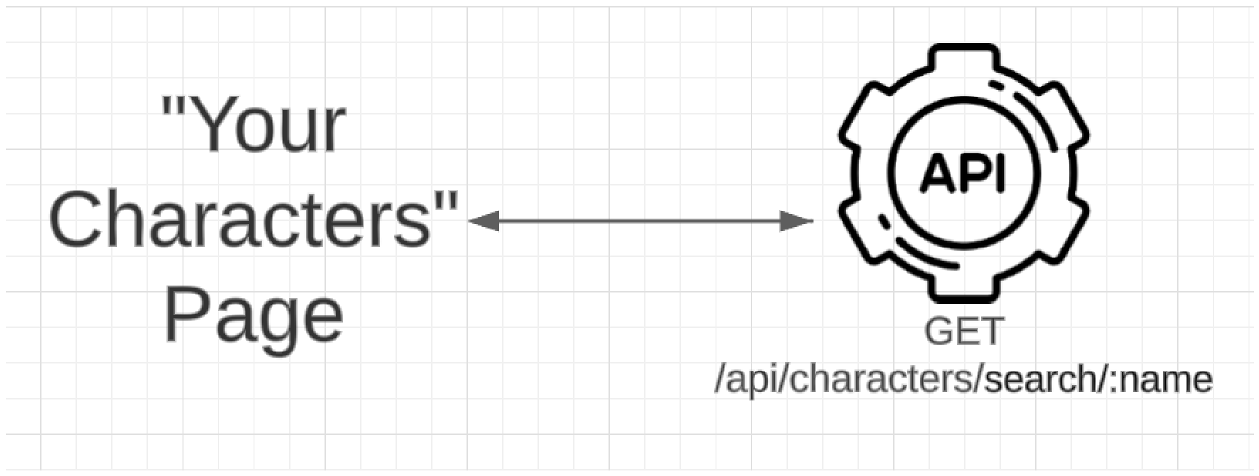
Success response:

```
{
  "name": "Jane Johnson",
 }
```

Error response:

```
{
  "message": "Character not found."
}
```

Diagram:



Edit/update a specific character (PUT):

URL: https://characternotebook.com/api/characters/:id

After a character has been created, users will be able to edit or update the information they previously provided about that character. By clicking the Edit button, users will be able to update or change the character's attributes. This endpoint will allow for such changes to be made.

Sample request:

```
{
  "name": "Jesamine Johnson",
  "age": 25,
  "gender": "Female",
  "gender": "Female",
  "species”: “human",
  “eyes”: “blue”,
 “hair”: “blonde and long”
}
```

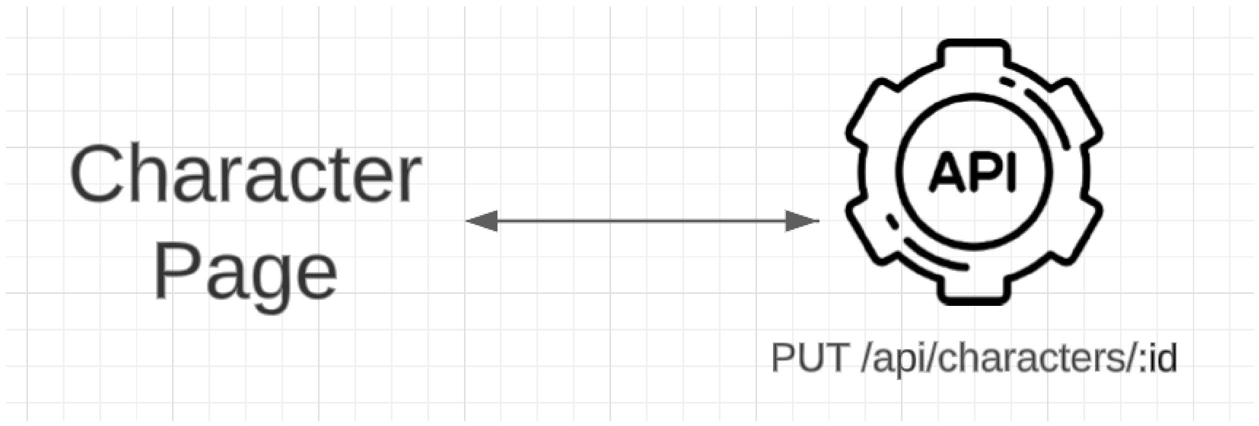
Success response:

```
{
  "message": "Character updated."
}
```

Error response:

```
{
  "error": "Character not found."
}
```

Diagram:



Delete a specific character (DELETE):

URL: https://characternotebook.com/api/characters/:id

Sometimes after creating a character, users may decide that they no longer like, want, or need a character any longer. Because of this, there will be the option for users to delete characters from their profile. This endpoint will be used to remove a character’s profile from the database.

Sample request:

DELETE https://characternotebook.com/api/characters/:id

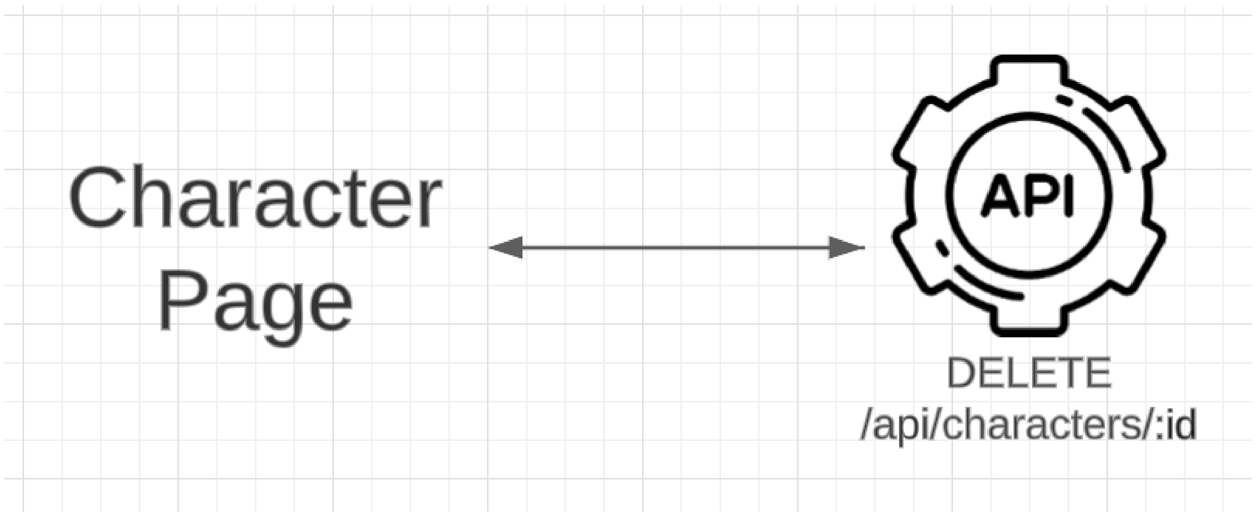
Success response:

```
{
  "message": "Character deleted."
}
```

Error response:

```
{
  "error": "Character not found."
}
```

Diagram:



Relationships

Create a new relationship between characters (POST):

URL: https://characternotebook.com/api/relationships/newrelationship

After creating at least two characters, users will be able to dictate what kind of relationship exists between characters. This is done by selecting two characters and the relationship type, which will be either friends, family, romance, or enemies. This endpoint will be used for this process.

Sample request:

```
{
  "character1name": “Jane Johnson”,
  "character2name": “Mike Michaelson”,
  "relationshipType": "Friends"
}
```

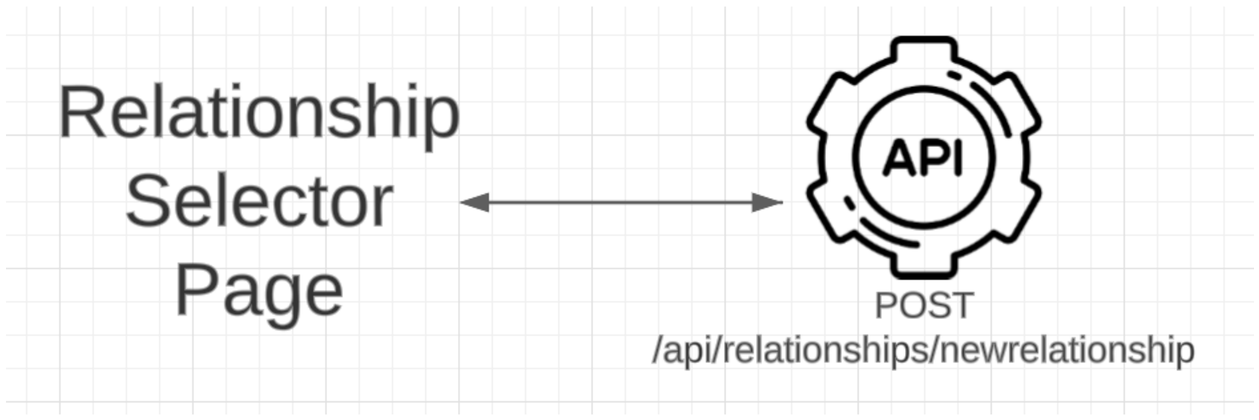
Success response:

```
{
  "message": "Relationship created.",
  "relationshipId": 789
}
```

Error response:

```
{
  "error": "Invalid characters."
}
```

Diagram:


Relationship Selector Page
API
POST /api/relationships/newrelationship

View existing relationships (GET):

URL: https://characternotebook.com/api/relationships/

After creating some relationships between characters, users will be able to view the existing relationships in a list or a visualized map. This endpoint is responsible for retrieving and displaying such information.

Request response:

GET https://characternotebook.com/api/relationships/

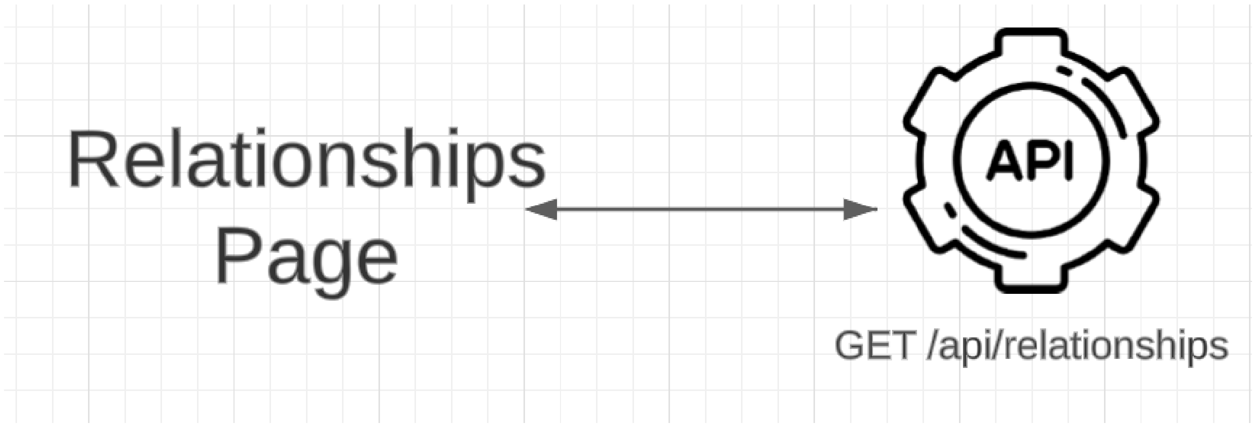
Success response:

```
{
  "relationships": [
    {
      "id": 1,
      "character1_id": 2,
      "character2_id": 3,
      "relationship_type": "Friends"
    },
    {
      "character1name": “Jane Johnson”,
      "character2name": “Mike Michaelson”,
      "relationship_type": "Friends"
    }
   {
      "character1name": “Jane Johnson”,
      "character2name": “Eliza Elsebeth”,
      "relationship_type": "Family"
    }
  ]
}
```

Error response:

```
{
  "message": "No relationships found. Create relationships to view them."
}
```

Diagram:



View details about a specific relationship (GET):

URL: https://characternotebook.com/api/relationships/:id

Sometimes users might prefer a more narrow approach to viewing relationship details. Because of this, there will be the option to view the relationship data pertaining to just one character. This endpoint is responsible for fetching relationship data for just the specified character.

Sample request:

GET https://characternotebook.com/api/relationships/:id

Success response:

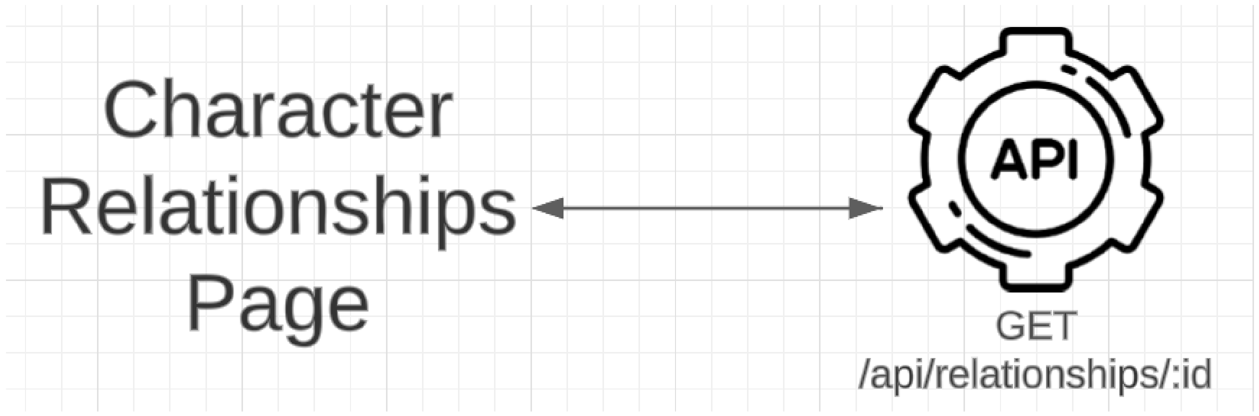
```
{
  "relationshipId": 456,
  "character1name": “Jane Johnson”,
  "character2name": “Mike Michaelson”,
  "relationshipType": "Friends"
```

}

Error Response:

```
{
  "error": "Relationship not found."
}
```

Diagram:



Update relationship information (PUT):

URL: https://characternotebook.com/api/relationships/:id

Users will have the ability to edit or update relationship details between two characters. This endpoint will handle relationship-related update requests pertaining to characters.

Sample request:

{

```
  "relationshipType": "Romance"
}
```

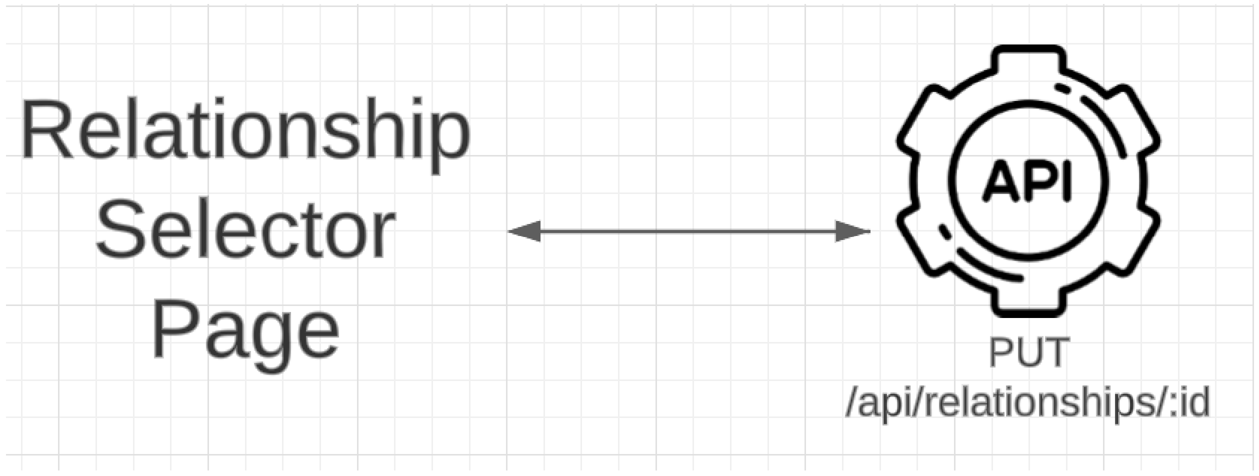
Success response:

```
{
  "message": "Relationship updated."
}
```

Error Response:

```
{
  "error": "Relationship not found."
}
```

Diagram:



Delete a relationship (DELETE):

URL: https://characternotebook.com/api/relationships/:id

Users may decide they no longer require or want a specific relationship to exist anymore. For this reason, there will be the option to delete relationships, and this endpoint will be responsible for removing relationship data from the database.

Sample request:

```
DELETE https://characternotebook.com/api/relationships/:id
```

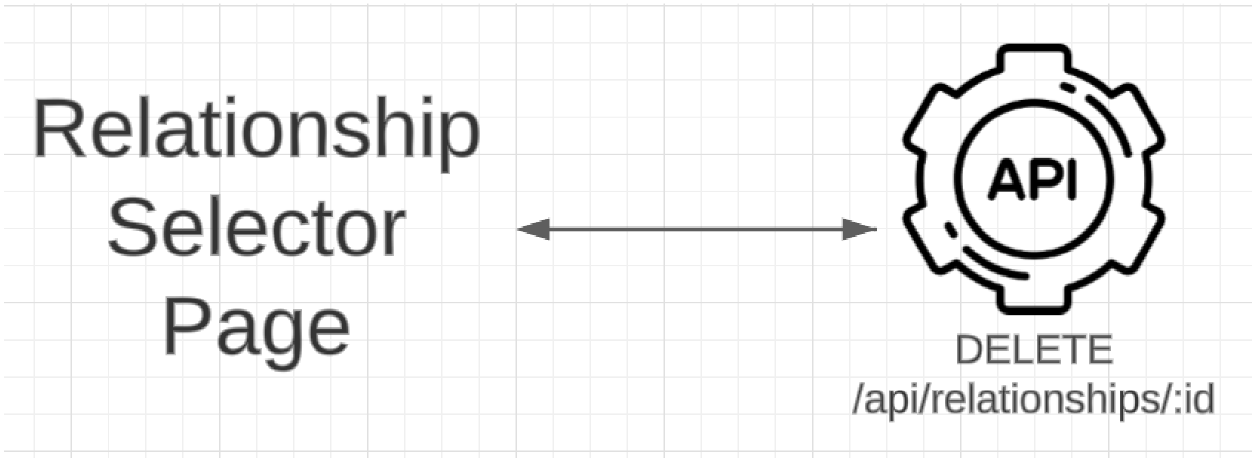
Success response:

```
{
  "message": "Relationship deleted."
}
```

Error response:

```
{
  "error": "Relationship not found."
}
```

Diagram:


Relationship Selector Page
API
DELETE /api/relationships/:id