Ensemble

A collection of all of the Ensemble endpoints.

Projects

POST Create a new Project

localhost:3000/api/projects

POST /api/projects

REQUIRES: title (string, Not null) description (text, not null) author (string, not null) project_num (string, not null, less than 72 characters) token (Logged In User's Token)

RETURNS: JSON Formatted message id: Project-ID title: Project Title description: Project Description author: Project Author

id: User-ID email: User email first_name: User first name last_name: User last name

ERROR MESSAGES: Will return error messages if a required field isn't filled out.

HEADERS

Content-Type

application/json

BODY

```
{
    "title" : "Title",
    "description" : "Description",
    "author" : "Author",
    "project_num" : "ProjectNum",
```

```
    "token" : "UserToKen"
}
```

Sample Request

Create a new Project

```
curl --request POST \
  --url localhost:3000/api/projects \
  --header 'content-type: application/json' \
  --data '{
    "title" : "Title",
    "description" : "Description",
    "author" : "Author",
    "project_num" : "ProjectNum",
    "token" : "UserToKen"
}'
```

PATCH Edit a Project

localhost:3000/api/projects/:id?token=current_usertoken

PATCH /api/projects/:id?token=:token

REQUIRES: :id (Project ID) :token (Project Owner's Token)

OPTIONAL: title: (Project Title, string) description: (Project Description, string) author: (Publication Author, string) project_num: (Project Number, string)

RETURNS: JSON Formatted message { project_id: Project ID, token: Project Token, description: Project Description, title: Title, author: Author for Project, project_num: Project Number owner: { id: User ID first_name: Owner Name last_name: Owner Last Name }, members: [{ "id": Member ID, "first_name": "Member First Name", "last_name": "Member Last Name" }], comments: ["left_comments": [{ "id": "Comment ID", "body": "Comment Body", "user": { "id": User ID, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [{ "id": Comment ID, "body": "Comment Body", "user": { "id": User ID, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [{ "id": Comment ID, "body": "Comment Body", "user": { "id": User ID, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [] }] }]]] } } ERROR MESSAGES:

HEADERS

Content-Type

application/json

BODY

```
{
  "description": "description",
  "title": "title",
  "author": "author",
  "project_num": "project_num"
}
```

Sample Request

Edit a Project

```
curl --request PATCH \
  --url 'localhost:3000/api/projects/:id?token=current_usertoken' \
  --header 'content-type: application/json' \
  --data '{
  "description": "description",
  "title": "title",
  "author": "author",
  "project_num": "project_num"
}'
```

POST Invite a User to a Project

localhost:3000/api/projects/:token/invitations

POST localhost:3000/api/projects/:project_token/invitations

REQUIRES :project_token :email (E-mail address of User to share project with)

RETURNS: JSON formatted message: a) ":email has been added to Project successfully!" b) ":email does not have an account and will have to sign up."

This will e-mail the users after the above checks are performed inviting them to sign up or view their project.

HEADERS

Content-Type

application/json

BODY

```
{
    "email" : "invited_user@example.com"
}
```

Sample Request

Invite a User to a Project

```
curl --request POST \
  --url localhost:3000/api/projects/:token/invitations \
  --header 'content-type: application/json' \
  --data '{
    "email" : "invited_user@example.com"
}'
```

GET Show a Project

localhost:3000/api/projects/:id?token=:user_token

GET /api/projects/:id?token=:token

REQUIRES: :id (Project ID) :token (Logged In User's Token)

RETURNS: JSON Formatted message { project_id: Project ID, token: Project Token, description: Project Description, title: Title, author: Author for Project, project_num: Project Number owner: { id: User ID first_name: Owner Name last_name: Owner Last Name }, members: [{ "id": Member ID, "first_name": "Member First Name", "last_name": "Member Last Name" }], comments: ["left_comments": [{ "id": "Comment ID", "body": "Comment Body", "user": { "id": 1, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [{ "id": Comment ID, "body": "Comment Body", "user": { "id": User ID, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [{ "id": Comment ID, "body": "Comment Body", "user": { "id": User ID, "first_name": "User First Name", "last_name": "User Last Name" }, "comments": [] }] }] }]] } } ERROR MESSAGES:

HEADERS

Content-Type

application/json

Sample Request

Show a Project

```
curl --request GET \
  --url 'localhost:3000/api/projects/:id?token=%3Auser_token' \
  --header 'content-type: application/json'
```

DELETE Delete a project

localhost:3000/api/projects/:id

DELETE localhost:3000/api/projects/:id

REQUIRES: :token - Project Owner token

RETURNS: JSON Formatted message message: "Project deleted successfully."

ERRORS: Error messages if any

HEADERS

Content-Type

application/json

BODY

```
{ "token" : "Project Owner's token" }
```

Sample Request

Delete a project

```
curl --request DELETE \
  --url localhost:3000/api/projects/:id \
  --header 'content-type: application/json' \
  --data '{ "token" : "Project Owner'\''s token" }'
```

Users

POST Create a new User

localhost:3000/api/users

POST /api/users

REQUIRES: email (Unique, not null) first_name (not null) last_name (not_null) password (not null, less than 72 characters)

RETURNS: JSON Formatted message id: User-ID email: User email first_name: User first name last_name: User last name

ERROR MESSAGES:

HEADERS

Content-Type

application/json

BODY

```
{
    "email" : "User E-Mail",
    "first_name" : "User First Name",
    "last_name" : "User Last Name",
    "password" : "User Password",
    "token" : "Project Token (Optional)"

}
```

Sample Request

Create a new User

```
curl --request POST \
  --url localhost:3000/api/users \
  --header 'content-type: application/json' \
  --data '{
    "email" : "User E-Mail",
    "first_name" : "User First Name",
    "last_name" : "User Last Name",
    "password" : "User Password",
    "token" : "Project Token (Optional)"
```

POST Login a User

localhost:3000/api/login

POST /api/login

REQUIRES: email (Unique, not null) password (not null, less than 72 characters)

RETURNS: JSON Formatted message id: User-ID email: User email first_name: User first name last_name: User last name token: User Token

ERROR MESSAGES:

HEADERS

Content-Type

application/json

BODY

```
{
    "email" : "User E-mail",
    "password" : "User Password"
}
```

Sample Request

Login a User

```
curl --request POST \
  --url localhost:3000/api/login \
  --header 'content-type: application/json' \
  --data '{
    "email" : "User E-mail",
    "password" : "User Password"
}'
```

PATCH Update a User

localhost:3000/api/users/1

PATCH localhost:3000/api/users/:id

REQUIRES: :token - User token

OPTIONAL: :first_name (Updated first name, string) :last_name (Updated last name, string) :password (Updated password, < 72 characters, string) :email (Updated email, string)

RETURNS: JSON Formatted message id: User ID first_name: Updated First Name last_name: Updated Last Name email: Updated E-mail

ERRORS: Error messages if any

HEADERS

Content-Type

application/json

BODY

```
{   "email" : "da_amazing_daisha@example.com",
    "password" : "playground",
    "last_name" : "Kietzman",
    "token" : "6NLJrDDC85XzomTpY9SRRL6a" }
```

Sample Request

Update a User

```
curl --request PATCH \
  --url localhost:3000/api/users/1 \
  --header 'content-type: application/json' \
  --data '{ "email" : "da_amazing_daisha@example.com",
    "password" : "playground",
    "last_name" : "Kietzman",
    "token" : "6NLJrDDC85XzomTpY9SRRL6a" }'
```

DELETE Logout a User

localhost:3000/api/logout

DELETE /api/logout

REQUIRES: Do not send a token with the request

RETURNS: JSON Formatted message "Logout complete."

ERROR MESSAGES: N/A

HEADERS

Content-Type

application/json

Sample Request

Logout a User

```
curl --request DELETE \
  --url localhost:3000/api/logout \
  --header 'content-type: application/json'
```

DELETE Delete a user

```
localhost:3000/api/users/:id
```

DELETE localhost:3000/api/users/1

REQUIRES: :token - User token

RETURNS: JSON Formatted message id: User ID first_name: User First Name last_name: User Last Name message: "User successfully disabled"

ERRORS: Error messages if any

HEADERS

Content-Type

application/json

BODY

```
{ "token" : "Current_User's token" }
```