

Steps to Run the Django Application Server and start the API

1. Move to the directory "irsc", start the cmd or terminal now.
2. Start the virtual environment using `venv\Scripts\activate` (Windows)
3. Now move to directory using `"cd irsc"`
4. Folder location will look like this:
`((venv) C:\Users\Naman Vats\Desktop\irsc\irsc>`
5. Here you can see `python manage.py`
6. Now type `"python manage.py runserver"` in the terminal
7. By Default the Server will run on <http://127.0.0.1:8000/> (If you open this you will receive Page Not Found, and URL Paths listed as it is an API so we focus upon endpoints URL only).

Note: Admin Page URL: <http://127.0.0.1:8000/admin>

Credentials :

- Username: namanvats
- Password: nullbyte1997

1. List All Users Endpoint

This API call will give the list of all present users registered in the system.

- **URL**

`/api/v1/users`

- **Method**

`GET`

- **Success Response**

Code: 200 "OK"

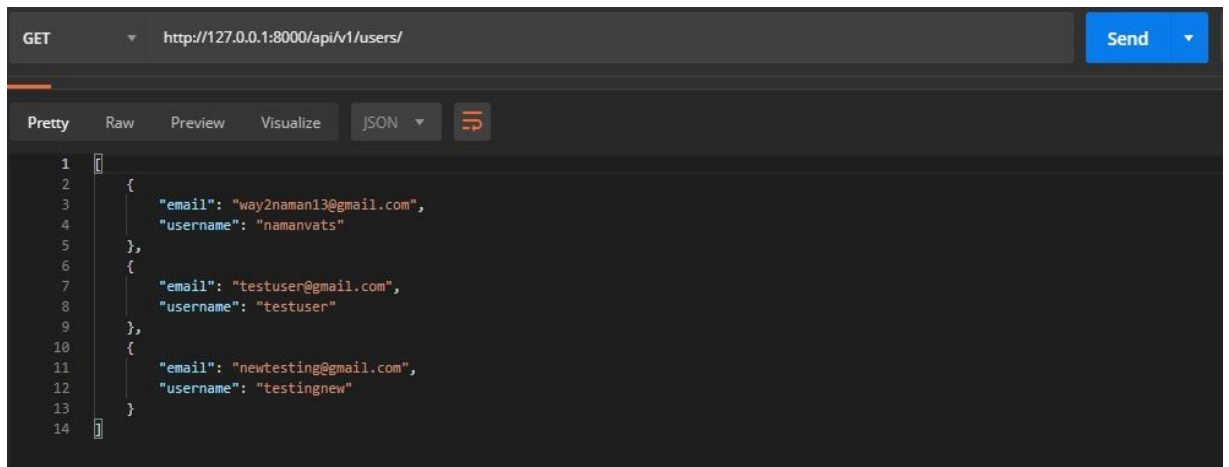
- **Error Response**

Code: 404 NOT FOUND

OR

Code: 401 UNAUTHORIZED

- **Sample Call**



```
GET http://127.0.0.1:8000/api/v1/users/ Send

Pretty Raw Preview Visualize JSON

1 {
2   {
3     "email": "way2naman13@gmail.com",
4     "username": "namanvats"
5   },
6   {
7     "email": "testuser@gmail.com",
8     "username": "testuser"
9   },
10  {
11    "email": "newtesting@gmail.com",
12    "username": "testingnew"
13  }
14 }
```

2. Login Endpoint

This call will allow the registered user to login, using username/email and password as parameters.

- **URL:**
 - `api/v1/rest-auth/login/`
- **Method:**
 - POST
- **Success Response:**
 - Code: 200 "OK"
- **Data Params:**
 - **username/email**
 - **password**
- **Error Response:**
 - Code: 400 "Bad Request"
 - `{"Unable to log in with provided credentials."}`
 - `{"This field is required."}`
 - Code: 403 "Forbidden"
 - `{"CSRF Failed: CSRF token missing or incorrect."}`

- **Sample Call: LOGIN**

Login

OPTIONS

Check the credentials and return the REST Token
If the credentials are valid and authenticated.
Calls Django Auth login method to register User ID
in Django session framework

Accept the following POST parameters: username, password
Return the REST Framework Token Object's key.

GET /api/v1/rest-auth/login/

HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

```
{  
  "detail": "Method \"GET\" not allowed."  
}
```

Raw data

HTML form

Username

Email

Password

POST

POST

http://127.0.0.1:8000/api/v1/rest-auth/login/

Send

Params

Authorization

Headers (10)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> username	namanvats	
<input checked="" type="checkbox"/> password		
Key	Value	Description

Body

Cookies (2)

Headers (9)

Test Results

Status: 200 OK Time: 617ms Size: 500 B Save

Pretty

Raw

Preview

Visualize

JSON

```
1 {  
2   "key": "15d76ad4f0fdbb520c476bcbe8e1c18081a5eb3a"  
3 }
```

3. Registration Endpoint

This call will allow the user to register, using username, password and email as parameters.

- **URL:**
 - /api/v1/rest-auth/registration/
- **Method:**
 - POST
- **Success Response:**
 - Code: 201 "Created"
- **Data Params:**
 - **username**
 - **email**
 - **password1**
 - **Password2** (confirming password1 both needs to be same)
- **Error Response:**
 - Code: 400 "Bad Request"

```
■ A user with that username already exists.  
■ A user is already registered with this e-mail address.  
■ This password is too short. It must contain at least 8 characters.
```

- **Sample Call:**

Creating New User

Untitled Request Comments 0

POST http://127.0.0.1:8000/api/v1/rest-auth/registration/ Send Save

Params Authorization Headers (10) **Body** Pre-request Script Tests Settings Cookies Code

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	newuser2			
<input checked="" type="checkbox"/>	email	newuser2@gmail.com			
<input checked="" type="checkbox"/>	password1	alphaheading			
<input checked="" type="checkbox"/>	password2	alphaheading			
	Key	Value	Description		

Body Cookies (3) Headers (10) Test Results Status: 201 Created Time: 1395ms Size: 849 B Save Response

Pretty Raw Preview Visualize JSON ⌵

```
1 {
2   "key": "3c2074e4db7c07d2c09bdc7f7Fa28a657ea5ca4e"
3 }
```

Creating User with Same Username (Bad Request)

POST http://127.0.0.1:8000/api/v1/rest-auth/registration/ Send Save

Params Authorization Headers (9) **Body** Pre-request Script Tests Settings Cookies Code

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	username	newuser2			
<input checked="" type="checkbox"/>	email	newuser2@gmail.com			
<input checked="" type="checkbox"/>	password1	alphaheading			
<input checked="" type="checkbox"/>	password2	alphaheading			
	Key	Value	Description		

Body Cookies Headers (7) Test Results Status: 400 Bad Request Time: 18ms Size: 354 B Save Response

Pretty Raw Preview Visualize JSON ⌵

```
1 {
2   "username": [
3     "A user with that username already exists."
4   ],
5   "email": [
6     "A user is already registered with this e-mail address."
7   ]
8 }
```

Page

The screenshot shows a REST client interface with a dark theme. At the top, the title 'Register' is displayed next to an 'OPTIONS' button. Below the title, the URL bar contains 'GET /api/v1/rest-auth/registration/'. The main response area shows an HTTP 405 error: 'HTTP 405 Method Not Allowed', with headers 'Allow: POST, OPTIONS', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON object: { "detail": "Method \"GET\" not allowed." }. Below the response, there are tabs for 'Raw data' and 'HTML form'. At the bottom, there is a form with four input fields: 'Username', 'Email', 'Password1', and 'Password2'. A 'POST' button is located at the bottom right of the form.

```
Register
```

OPTIONS

GET /api/v1/rest-auth/registration/

```
HTTP 405 Method Not Allowed
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept

{
  "detail": "Method \"GET\" not allowed."
}
```

Raw data HTML form

Username |

Email

Password1

Password2

POST

4. Status/Ping Endpoint

This API Endpoint will tell the status of the server,database,api whether they are up or not, if up it will return the status in JSON format true or pong.

- **URLs**
 - /status/
 - status/api/health/
- **Method**
 - POST
- **Success Response**
 - Code: 200 "OK"

- **Sample Call**

GET

http://127.0.0.1:8000/status/api/health

Send

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION	...
Key	Value	Description	

Body

Cookies

Headers (5)

Test Results

Status: 200 OK

Time: 8ms

Size: 260 B

Save

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

```
{
  "ping": {
    "pong": true
  },
  "databases": {
    "default": true
  },
  "caches": {
    "default": true
  }
}
```