# Project Overview & Release Notes

**Project Description**: This Django project is the solution for the Backend Engineer Case Study [Numbers](https://docs.google.com/document/d/1Qug-XhnFT8QOUi5VmrwZmRhUZFtqjXiG0OOmpXYdp7A/edit?usp=sharing) assignment. In summary, it’s a Python Django application that exposes two endpoints to convert any number given into the English words that describe that number.

**Repository**: <https://github.com/andiebalverde/numbers_to_english_django_n>

**Author**: Andres Balverde / Github: andiebalverde / Email: [andie.balverde@gmail.com](mailto:andie.balverde@gmail.com)

## Project Specifications and Limit of Scope:

**The Framework**: The particular implementation of Django that’s been used for this project is Django Ninja, a web framework for building APIs with Django and Python 3.6+. As main features, the framework is based on Django core, ORM, based on the OpenAPI open standards for APIs and Json Schema

**Limit of Scope**: as introduced in the “Project Description” above, the goal is to convert any given number into English words that describe the number. This MVP version of the solution is limited to positive integers below quadrillion.

# Instructions to Run the Project:

This section is intended for coworkers or team leaders to run the project in their local environments.

1. **Clone Project**: positioned on a base folder, run the following command using git bash (or other terminal application with emulation layer for Git commands)

Note: previously request access to the repository for the owner to send a collaboration invite.

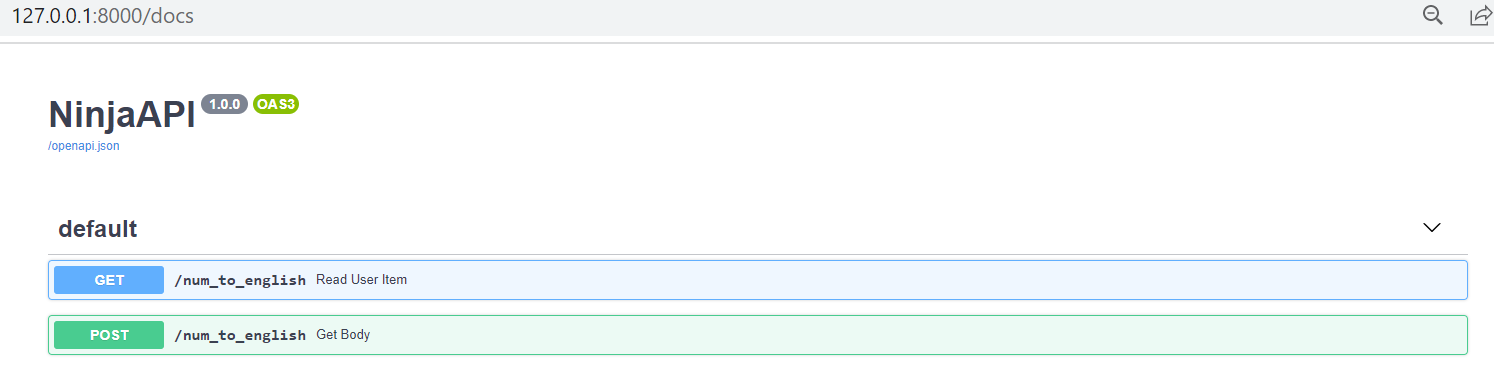
git clone https://github.com/andiebalverde/numbers\_to\_english\_django\_n.git

This will create a project folder and copy a ‘main’ version of the project locally.

1. **Install Requirements**: cd to the project folder and run the command ‘pip install –r requirements.txt’ to install the necessary libraries for the project.

Note: pip is a standard package manager used to install and maintain python packages. See [How to Install PIP on Windows](https://www.geeksforgeeks.org/how-to-install-pip-on-windows/#:~:text=Download%20and%20Install%20pip%3A&text=Download%20the%20get%2Dpip.py,where%20the%20above%20file%20exists.&text=and%20wait%20through%20the%20installation,now%20installed%20on%20your%20system.) if needed

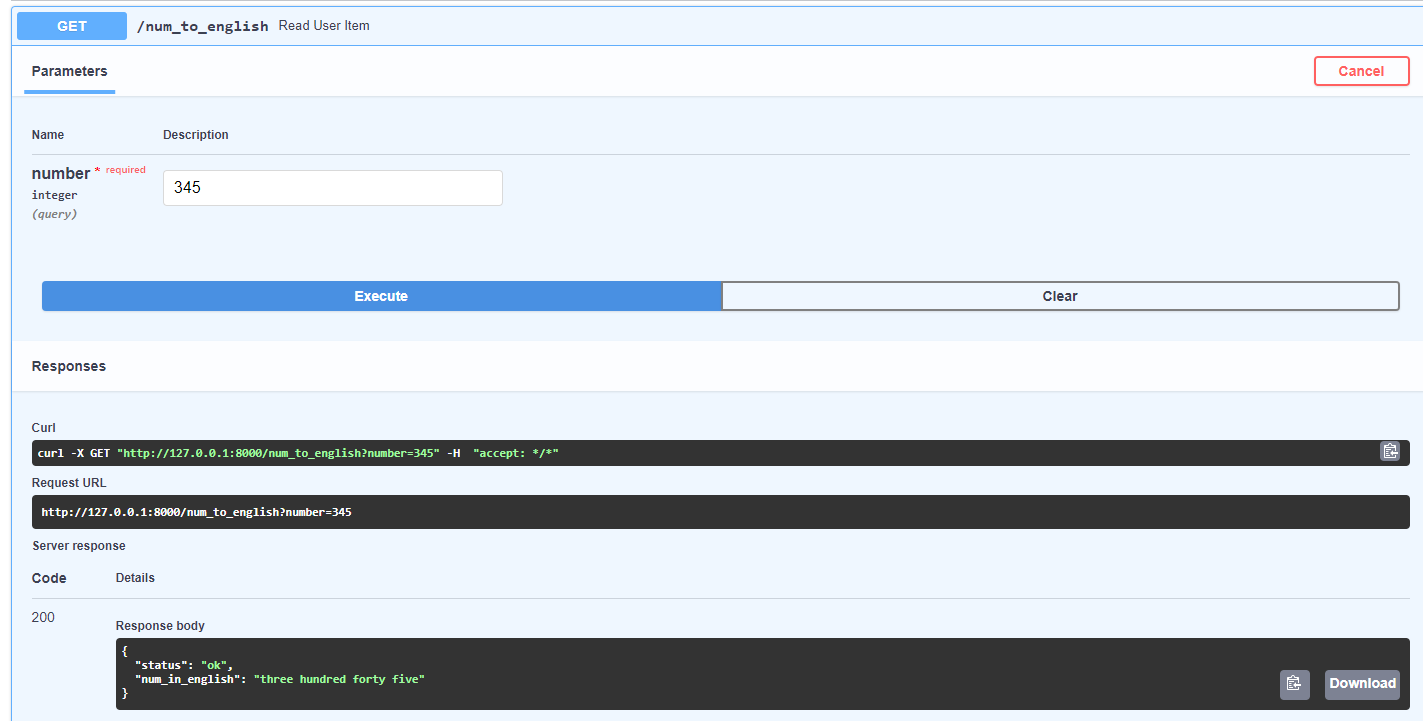
1. **Run Project in Local Host**: cd to djninja folder and run the command ‘python manage.py runserver’. This will start development server (ie: at <http://127.0.0.1:8000>
2. **Open Interactive Documentation**: browse to <http://127.0.0.1:8000/docs> which will open interactive documentation based on the open standards for APIs: OpenAPI and JSON Schema. Here, you should be able to see the available endpoints as:



# Instructions to Test the Endpoints:

## Get Endpoint Test:

Using the OpenAPI interface (<http://127.0.0.1:8000/docs>), click on the ‘GET’ button, click on the ‘Try it Out’ button, specify an integer and press the ‘Execute’ button. This is how the Response should look like:



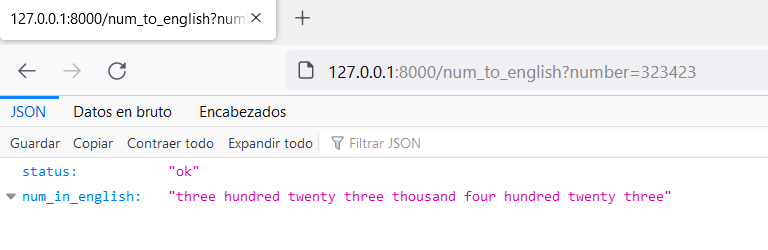
As specified, the response body includes ‘status’ (which could be “ok” or “fail”). And for “ok” status, the “num\_in\_english” field provides the English phrase that describes the given integer.

This API receives an HTTP GET request at /num\_to\_english, takes, validates and type-casts the parameters\*, decodes the result to JSON

generates an OpenAPI schema for defined operation.

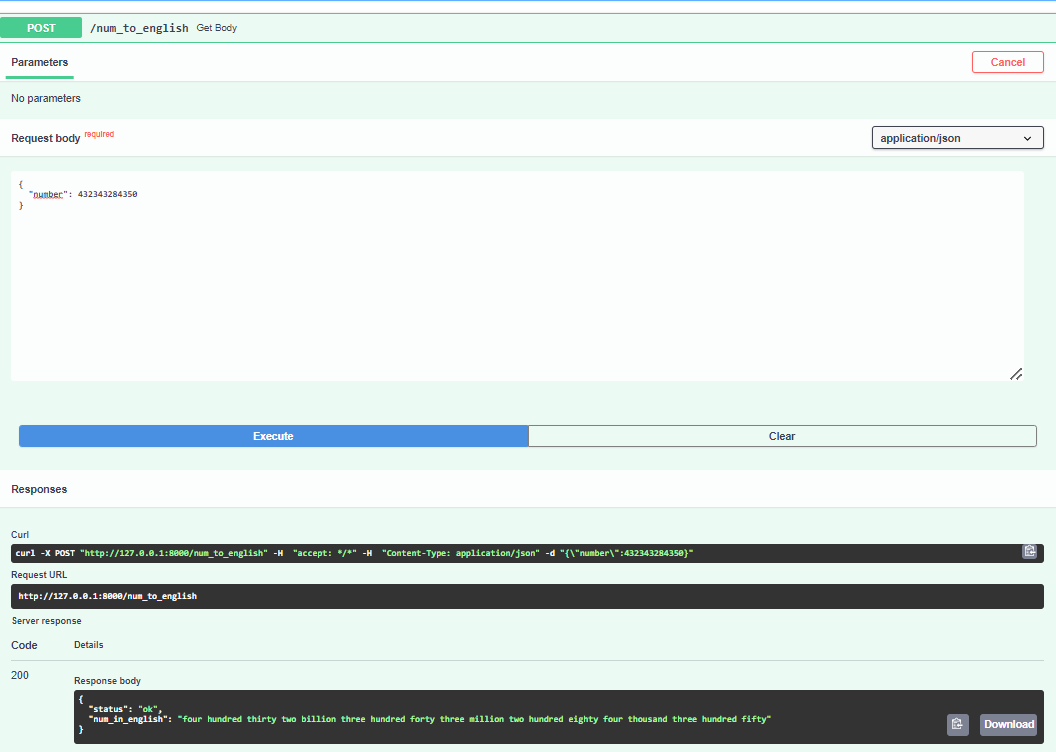
Since the \*parameter is declared to be of the type of Pydantic BaseModel, it will be interpreted as a request body, in line with the specs for the Get endpoint.

Alternatively, the following url including query parameter ‘number’ and the desired integer, will produce the same outcome:



## Post Endpoint Test:

Back to the OpenAPI interactive documentation (<http://127.0.0.1:8000/docs>), click on the ‘POST’ button, click on the ‘Try it Out’ button, specify a positive integer in the Request Body and press the ‘Execute’ button. This is how the Response should look like:



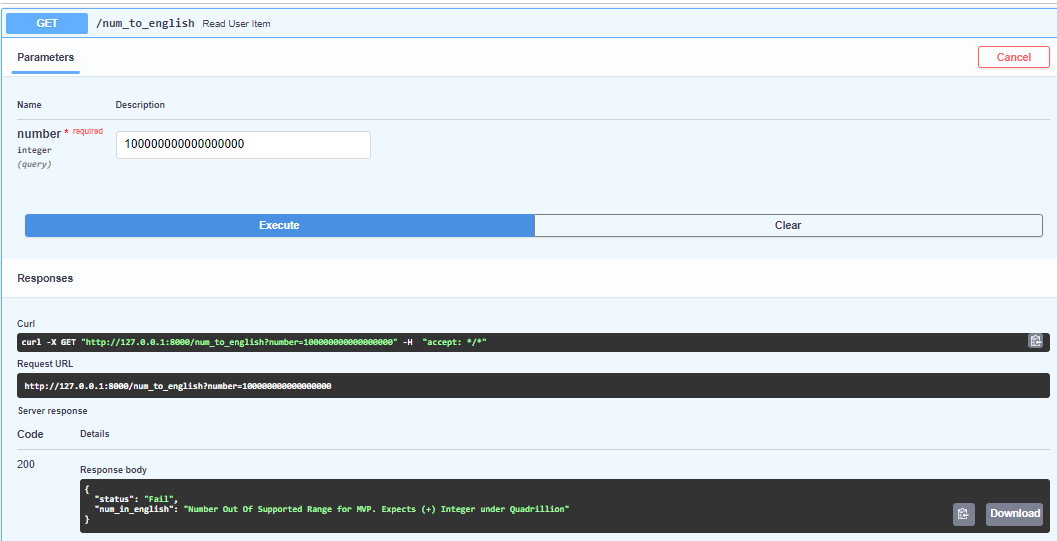
# Error Handling:

Status is reserved for messaging back if the process succeeded or failed. This MVP version takes care of the following errors:

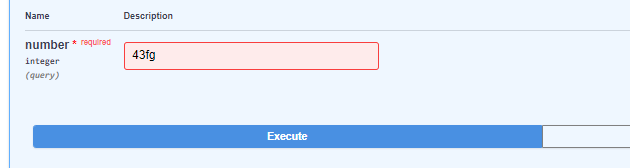
- Negative integers

- Integers above quadrillion.

For example, having a positive integer above quadrillion:



Note: Pydantic Model takes care of the error where the input number is not an integer. It requires the number field to be an integer and doesn’t allow the user to execute a request with a non-integer value. Hence, non-integer values are already handled by the Pydantic based request setup, not needing to be handled by the logic.



A red highlight of the field after the Execute button is pressed, indicates the value is not an integer.