# Flask Name Entity Recognition Application Documentation

## Over View:

This Flask application provides a web interface for performing Named Entity Recognition (NER) using spaCy, with the ability to evaluate the model's performance against ground truth annotations. The application supports visualization of identified entities and calculates various performance metrics.

## Dependencies:

 Flask: Web framework
 spaCy: NLP library for entity recognition
 scikit-learn: For calculating evaluation metrics
 Python-Markdown: For Markdown processing
 Python 3.12

Html, Css, JS - For Making Responsive Website

## Core Components

**Main Application (app.py)**

The main Flask application file that handles routing and integrates all components.

**Functions**

**calculate_metrics(pred_entities, true_entities, text_length)**

Calculates performance metrics for NER predictions.

Parameters:

- pred_entities: List of predicted entity dictionaries
- true_entities: List of ground truth entity dictionaries
- text_length: Length of the input text

Returns: Dictionary containing:

- precision
- recall
- f1_score
- accuracy

All metrics are rounded to 3 decimal places.

**extract_entities(doc)**

Extracts entities from a spaCy document.

Parameters:

- doc: spaCy Doc object

Returns: List of dictionaries containing:

- text: Entity text
- label: Entity label
- start: Starting character position
- end: Ending character position

**parse_ground_truth(text, entities_json)**

Parses and validates ground truth entity annotations.

Parameters:

- text: Original input text
- entities_json: JSON string containing ground truth entities

Returns: List of validated entity dictionaries

## Routes

**GET /**

Serves the main application page.

**POST /extract**

Handles NER processing and evaluation.

Form Parameters:

- rawtext: Input text for entity recognition
- ground_truth: Optional JSON string containing ground truth entities

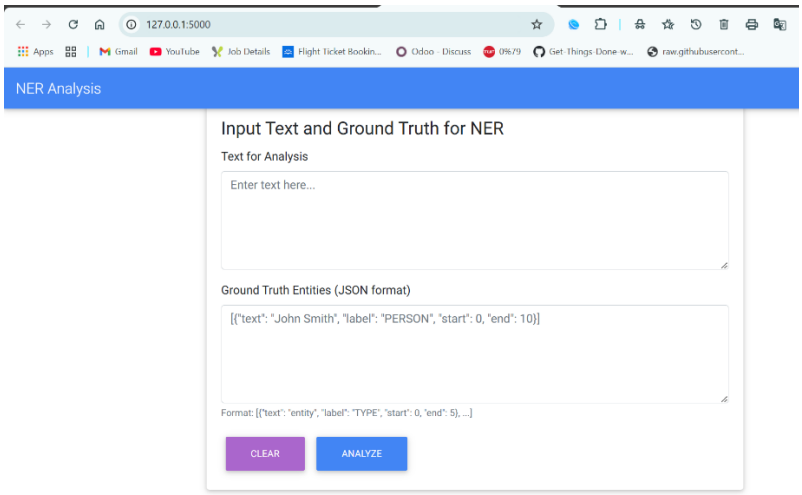
Returns: Rendered template with:

- Original text
- Visualized entities
- Performance metrics
- Predicted entities
- Ground truth entities

## *Usage Examples*

### Basic Entity Recognition

1. Navigate to the homepage
2. Enter text in the input field
3. Submit without ground truth to see entity predictions




---

## *Evaluation with Ground Truth*

1. Navigate to the homepage
2. Enter text in the input field
3. Provide ground truth annotations in JSON format, with **start_index and end_index** Format

For Example:"X.com merged with Confinity in 2000 to form PayPal. In 2002, Musk acquired United States citizenship, and that October eBay acquired PayPal for $1.5 billion. Using $100 million of the money he made from the sale of PayPal, Musk founded SpaceX, a spaceflight services company, in 2002.." This Is my original Sentence, Modify it in json format with start_index and end_index.

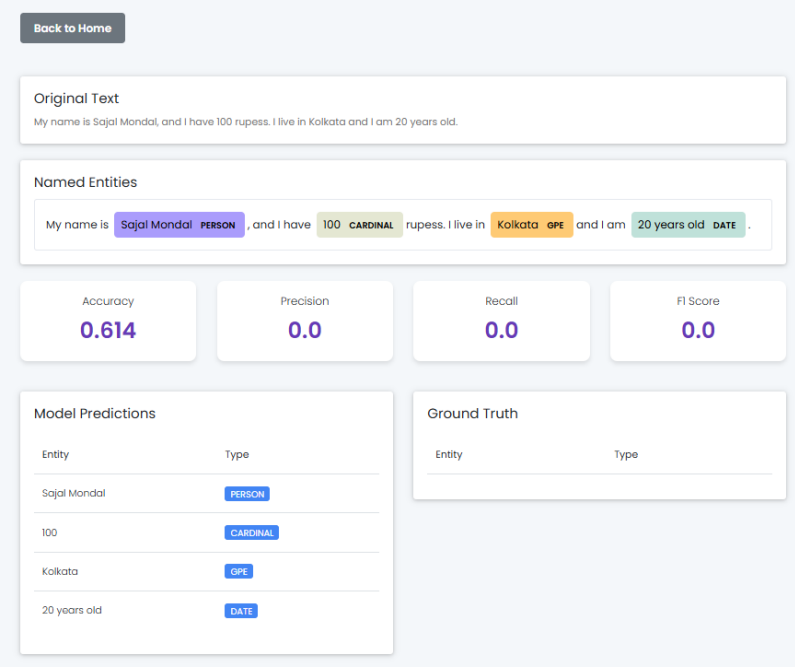

```
[
  {
    "text": "X.com",
    "label": "ORG",
    "start": 0,
    "end": 5
  },
  {
    "text": "Confinity",
    "label": "ORG",
    "start": 18,
    "end": 27
  },
  {
    "text": "2000",
    "label": "DATE",
    "start": 31,
    "end": 35
  },
  {
    "text": "PayPal",
    "label": "ORG",
    "start": 44,
    "end": 50
  },
  {
    "text": "2002",
    "label": "DATE",
    "start": 55,
    "end": 59
  },
```


The **start_index** index tells you where the entity begins in the string.

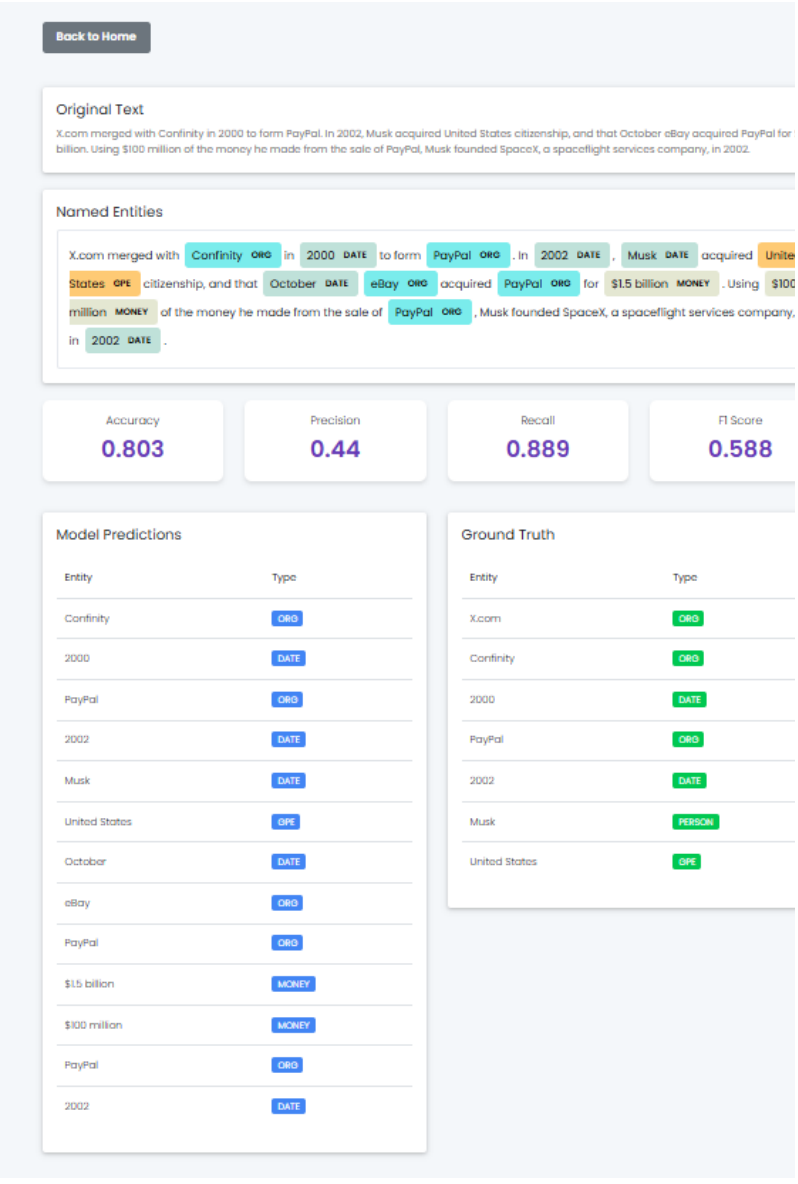
The **end_index** index tells you where the entity ends, just after the last character of the entity.

4. Submit to see predictions and evaluation metrics
5. IF there is no recongnition It will give the accuracy 0 and also you can see the precison , recall and F1 Score after giving the ground truth, otherwise it will not showing any value.
6. Accuracy also Dynamic, based on the result or correct answer it will show the accuracy.

## *Before add Ground truth*




## *After Adding Ground Truth*




## *Error Handling*

- Invalid ground truth JSON returns empty entity list
- Out-of-bounds entity indices are filtered out
- Non-matching entity text is filtered out

Missing required entity fields are filtered out

## *Performance Considerations*

- Uses spaCy's small English model (en_core_web_sm) for better performance
- Implements efficient binary classification for metric calculation
- Handles zero-division cases in metric calculations