FINANCIAL NEWS ENTITY EXTRACTION

A PROJECT REPORT

For

“SEC162 : GENERATIVE AI”

Submitted by

Y Dedeepya[Reg. No:AP23110011375]

P Evangeline[Reg. No:AP23110011389]

Y Sahithi[Reg. No:AP23110011384]

K. Datta Supriya[Reg. No:AP23110010648]

Tasneem Shaik[Reg. No:AP23110011575]

SEM: V

SECTION: S

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

SCHOOL OF ENGINEERING AND SCIENCES

SRM University-AP, Neerukonda, Andhra Pradesh 522240

December 2025

Financial News Entity Extraction

Project Description:

The **Financial News Entity Extraction** system is an AI-powered application designed to automatically identify and classify important information from financial news articles. Using modern **Natural Language Processing (NLP)** techniques and a **deep learning based Named Entity Recognition (NER)** model, the system extracts key entities such as organizations, people, monetary values, dates, percentages, and other financial terms.

This tool is especially helpful for students, financial analysts, researchers, and business enthusiasts who want to quickly understand the core information in any financial article without manually reading the complete text. The solution uses **spaCy's pre-trained NER model**, deployed through a **Flask web application**, and provides a clean, user-friendly interface where users can paste or browse financial news.

Project Scenarios:

Scenario 1: Financial Market Study

A finance student preparing a report wants to analyze multiple financial news articles. Instead of reading each article thoroughly, the student pastes the news text into the system. The application automatically highlights important entities like interest rates, company names, central banks, and key financial numbers making the research process faster and more accurate.

Scenario 2: Investor Quick Insights

An investor wants to monitor updates related to companies such as Apple, Tesla, or Microsoft. By feeding news articles into the system, they instantly obtain structured information such as stock-related numbers, executive names, earning dates, and announcements. This allows the investor to make faster financial decisions.

Scenario 3: Financial News Monitoring for Analysts

Market analysts who receive dozens of financial updates daily can simply paste the incoming news into the application. The system extracts relevant financial indicators (interest rates, inflation numbers, yield percentages, etc.), helping them quickly assess the economic impact without detailed manual reading.

Prerequisites:

To build and execute this project, the following tools and packages are required:

Software

- Python 3.x
- Anaconda Navigator / VS Code
- Web browser (Chrome recommended)

Python Packages:

- pip install spacy flask pandas numpy
- python -m spacy download en_core_web_sm

Prior Knowledge:

You should be familiar with:

- Basic Python programming
- Machine Learning fundamentals
- NLP concepts like tokenization and named entity recognition
- Flask routes and API requests
- HTML & JavaScript basics

Project Flow

- User enters or selects a financial news article
- Text is sent to the backend using a POST request
- The spaCy NLP model processes the text
- The NER model detects and classifies entities
- Extracted entities are returned as JSON
- UI displays highlighted text and categorized entity list

Project Activities

1. Data Collection & Preparation

- Collected a financial news dataset containing articles related to companies, markets, and economic events
- Loaded the dataset from CSV format
- Cleaned and preprocessed the text data (handling missing values, text normalization)

- Prepared textual data for NLP-based Named Entity Recognition

2. Exploratory Data Analysis

- Studied the structure and characteristics of the financial news dataset
- Performed descriptive analysis on article length and content
- Analyzed the frequency of different named entities
- Observed patterns of financial entities across multiple articles

3. Named Entity Recognition Implementation

- Loaded a pre-trained spaCy English language model
- Implemented Named Entity Recognition to extract financial entities
- Identified entities such as organizations, persons, monetary values, dates, and locations
- Applied NER on individual articles and dataset-level text

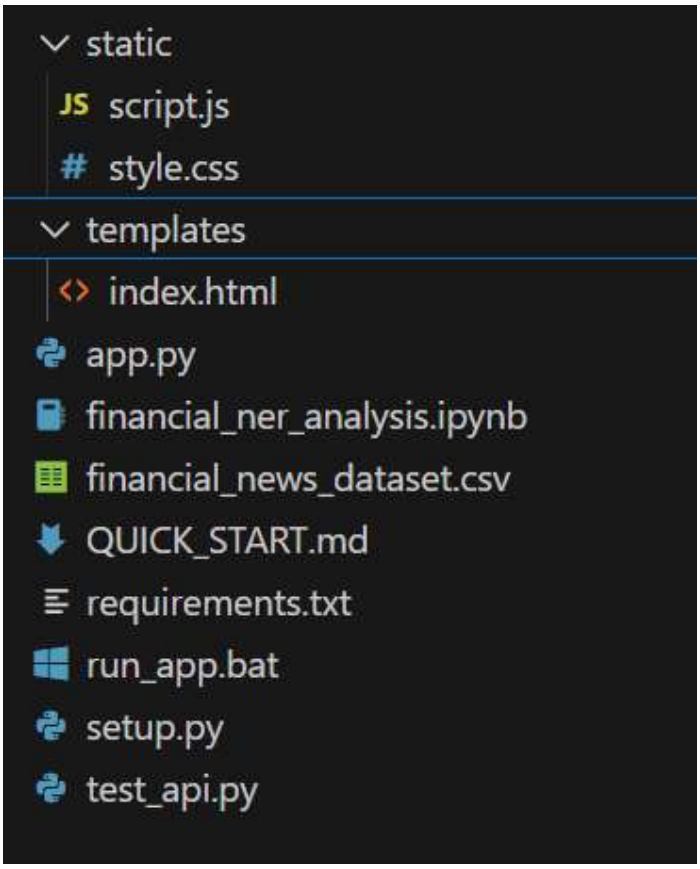
4. Result Analysis & Validation

- Analyzed extracted entities for correctness and relevance
- Compared entity extraction results across different news articles
- Verified consistency of recognized financial entities
- Documented observations and limitations of the model

5. Documentation & Reporting

- Documented dataset description, methodology, and results
- Prepared project report explaining NER workflow and findings
- Included screenshots and outputs from Jupyter Notebook execution

Project Structure:
Create the Project folder which contains files as shown below



Project Structure Explanation

- **static/**
  Contains static assets used for the user interface
  - **style.css –** Styling for the web interface
  - **script.js –** JavaScript for client-side interactions

- **templates/**
  Contains HTML templates
  - **index.html** – Main HTML page for displaying the interface

- **app.py**
  Python script for implementing the Named Entity Recognition logic and handling application flow

- **financial_ner_analysis.ipynb**
  Jupyter Notebook containing data loading, preprocessing, and Named Entity Recognition implementation

- **financial_news_dataset.csv**
  Dataset containing financial news articles used for entity extraction

- **requirements.txt**
  List of Python libraries and dependencies required to run the project

- **QUICK_START.md**
  Instructions for quickly understanding and running the project workflow

- **run_app.bat**
  Batch file to execute the Python application on Windows

- **setup.py**
  Project setup and configuration script

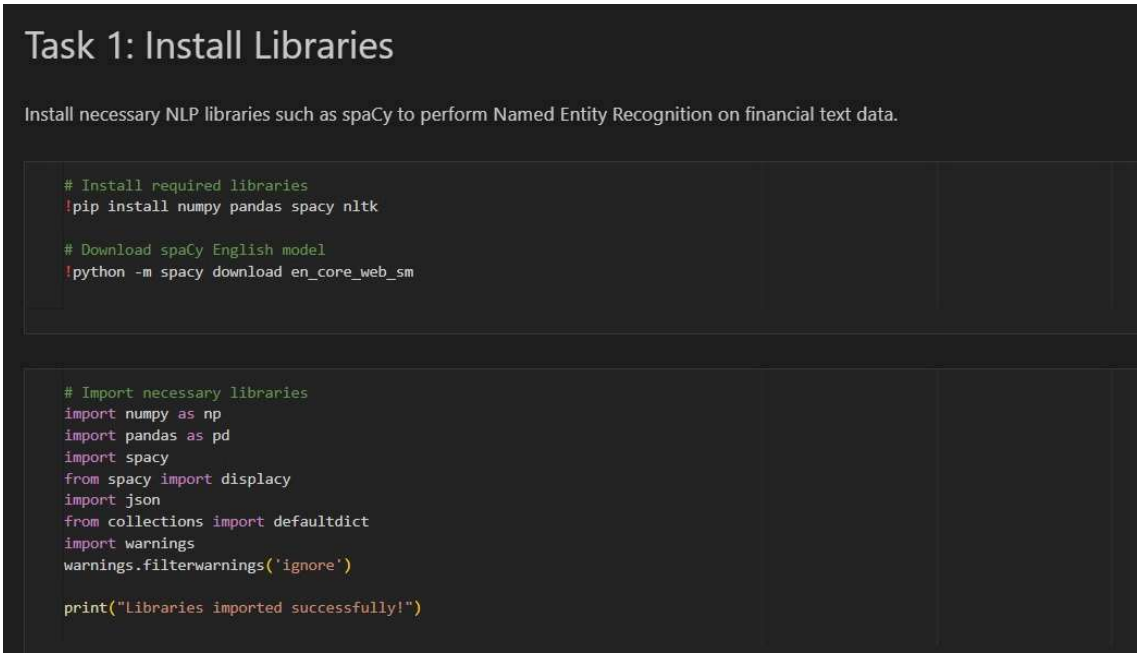
- **test_api.py**
  Script used for testing the functionality of the NER processing logic

Milestone 1: Data Collection and Preparation

In this milestone, we collected sample financial news articles and prepared the data for NLP processing. Although the project uses a pre-trained NER model, the dataset helped in testing and validating the model outputs.

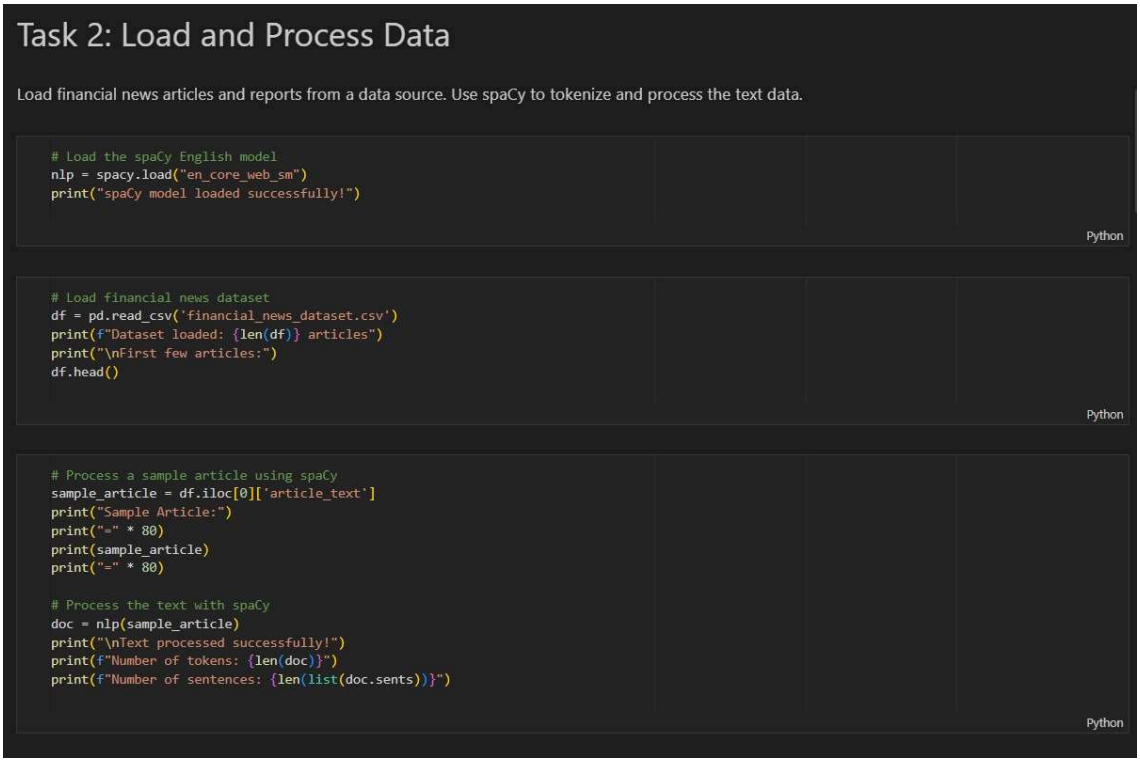
Activity 1.1 - Importing Required Python Libraries

We imported important libraries such as **spaCy**, **pandas**, **numpy**, and **Flask**. These libraries form the core infrastructure for NLP processing and backend development.



Activity 1.2 - Loading the Financial News Dataset

A sample dataset containing titles, dates, and news articles was loaded using pandas. This allowed us to inspect the variety of financial terms present.



Activity 1.3 - Data Cleaning and Preprocessing

We performed light preprocessing because spaCy's pipeline automatically handles tokenization and text normalization. We checked for:

- Duplicate articles
- Missing values
- Encoding issues

Activity 1.4 - Exploratory Data Analysis (EDA)

Before model implementation, we analyzed:

- Article word counts

- Frequency of common financial terms
- Occurrence of dates, organizations, and monetary values

Milestone 2: NLP Model Integration

Activity 2.1 - Loading the spaCy Pre-trained Model

We used the model en_core_web_sm, which contains deep learning architecture capable of recognizing multiple entity types.

Activity 2.2 - Implementing the NER Pipeline

The core extraction process was built inside the backend:
doc = nlp(text)
for ent in doc.ents:
print(ent.text, ent.label_)

This pipeline identifies:

- Organizations
- People
- Money values
- Dates
- Locations
- Percentages

Milestone 3: Backend Development (Flask API)

```
from flask import Flask, render_template, request, jsonify
import spacy
import pandas as pd
from collections import defaultdict
import json

app = Flask(__name__)

# Load spaCy model
try:
    nlp = spacy.load("en_core_web_sm")
    print("spaCy model loaded successfully!")
except OSError:
    print("Error: spaCy model 'en_core_web_sm' not found. Please run: python -m spacy download en_core_web_sm")
    nlp = None

# Load financial news dataset
try:
    df = pd.read_csv('financial_news_dataset.csv')
    print(f"Dataset loaded: {len(df)} articles")
except FileNotFoundError:
    print("Warning: financial_news_dataset.csv not found")
    df = None
except Exception as e:
    print(f"Warning: Error loading dataset: {e}")
    df = None


def perform_ner(doc):
    """
    Perform Named Entity Recognition on financial text using spaCy Doc object.

    Parameters:
```

```
    doc: spaCy Doc object containing processed text

    Returns:
    entities: Dictionary with entity labels as keys and lists of entities as values
    """
    entities = {}

    # Iterate through named entities in the document
    for ent in doc.ents:
        label = ent.label_
        text = ent.text

        # Initialize list for label if it doesn't exist
        if label not in entities:
            entities[label] = []

        # Append entity text to the corresponding list (avoid duplicates)
        if text not in entities[label]:
            entities[label].append(text)

    return entities


@app.route('/')
def index():
    """Render the main page"""
    return render_template('index.html')


@app.route('/analyze', methods=['POST'])
def analyze():
    """Analyze text input for named entities"""
    if nlp is None:
        return jsonify({'error': 'spaCy model not loaded. Please install it first.'}), 500

    data = request.get_json()
    text = data.get('text', '')

    # Log the user input (visible in terminal/console)
    print("\n" + "=" * 80)
    print("USER INPUT RECEIVED:")
    print("=" * 80)
    print(text)
    print("=" * 80 + "\n")

    if not text:
        return jsonify({'error': 'No text provided'}), 400

    # Process text with spaCy
    doc = nlp(text)

    # Perform NER
    entities = perform_ner(doc)

    # Format entities with additional information
    formatted_entities = []
    for label, entity_list in entities.items():
        for entity_text in entity_list:
            # Find the entity in the document to get start/end positions
            for ent in doc.ents:
                if ent.text == entity_text and ent.label_ == label:
                    formatted_entities.append({
                        'text': entity_text,
                        'label': label,
```

```
                    'description': get_entity_description(label),
                    'start': ent.start_char,
                    'end': ent.end_char
                })
                break

    return jsonify({
        'entities': formatted_entities,
        'entity_summary': {label: len(entity_list) for label, entity_list in entities.items()},
        'total_entities': len(formatted_entities)
    })
@app.route('/articles', methods=['GET'])
def get_articles():
    """Get list of articles from dataset"""
    if df is None:
        return jsonify({'error': 'Dataset not loaded'}), 500

    articles = df[['article_id', 'title', 'date']].to_dict('records')
    return jsonify({'articles': articles})
@app.route('/article/<int:article_id>', methods=['GET'])
def get_article(article_id):
    """Get specific article and perform NER on it"""
    if nlp is None:
        return jsonify({'error': 'spaCy model not loaded'}), 500
    if df is None:
        return jsonify({'error': 'Dataset not loaded'}), 500
    article = df[df['article_id'] == article_id]
    if article.empty:
        return jsonify({'error': 'Article not found'}), 404
    article_text = article.iloc[0]['article_text']
    title = article.iloc[0]['title']
    date = article.iloc[0]['date']
```

```
    # Process with spaCy
    doc = nlp(article_text)
    # Perform NER
    entities = perform_ner(doc)
    # Format entities
    formatted_entities = []
    for label, entity_list in entities.items():
        for entity_text in entity_list:
            for ent in doc.ents:
                if ent.text == entity_text and ent.label_ == label:
                    formatted_entities.append({
                        'text': entity_text,
                        'label': label,
                        'description': get_entity_description(label),
                        'start': ent.start_char,
                        'end': ent.end_char
                    })
                    break
    return jsonify({
        'article_id': int(article_id),
        'title': title,
        'date': date,
        'text': article_text,
        'entities': formatted_entities,
        'entity_summary': {label: len(entity_list) for label, entity_list in entities.items()},
        'total_entities': len(formatted_entities)
    })
def get_entity_description(label):
    """Get human-readable description for entity labels"""
    descriptions = {
        'PERSON': 'Person',
```

```
        'PERSON': 'Person',
        'ORG': 'Organization',
        'GPE': 'Geopolitical Entity',
        'LOC': 'Location',
        'MONEY': 'Monetary Value',
        'DATE': 'Date',
        'TIME': 'Time',
        'PERCENT': 'Percentage',
        'CARDINAL': 'Cardinal Number',
        'ORDINAL': 'Ordinal Number',
        'PRODUCT': 'Product',
        'EVENT': 'Event',
        'LAW': 'Law',
        'LANGUAGE': 'Language',
        'WORK_OF_ART': 'Work of Art',
        'FAC': 'Facility',
        'NORP': 'Nationalities or Religious/Political Groups',
        'QUANTITY': 'Quantity'
    }
    return descriptions.get(label, label)

@app.route('/test-api', methods=['GET'])
def test_api():
    """Test endpoint to see API response format"""
    sample_text = "Apple Inc. (AAPL) reported $89.5 billion in revenue. CEO Tim Cook announced the results."
    doc = nlp(sample_text) if nlp else None
    if doc:
        entities = perform_ner(doc)
        formatted_entities = []
        for label, entity_list in entities.items():
```

```
            for entity_text in entity_list:
                for ent in doc.ents:
                    if ent.text == entity_text and ent.label_ == label:
                        formatted_entities.append({
                            'text': entity_text,
                            'label': label,
                            'description': get_entity_description(label),
                            'start': ent.start_char,
                            'end': ent.end_char
                        })
                        break
        return jsonify({
            'message': 'Sample API Response',
            'sample_text': sample_text,
            'entities': formatted_entities,
            'entity_summary': {label: len(entity_list) for label, entity_list in entities.items()},
            'total_entities': len(formatted_entities)
        })
    else:
        return jsonify({'error': 'spaCy model not loaded'})
if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

Activity 3.1 - Creating the Flask Server

- Developed a Flask-based backend to process user input text
- Integrated spaCy NLP pipeline for Named Entity Recognition
- Implemented logic to extract and organize financial entities
- Verified backend functionality through local execution

Activity 3.2 – API Endpoint Implementation

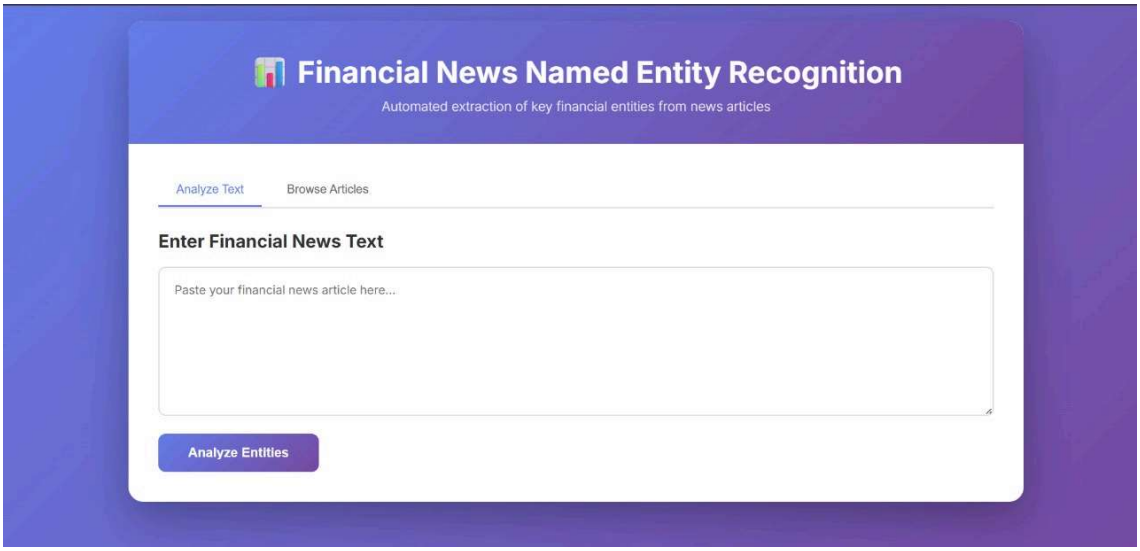
- Implemented an API endpoint to accept input text
- Processed input using spaCy NLP model
- Returned extracted named entities grouped by category
- Validated API output using test scripts

Milestone 4: Frontend Development

Activity 4.1 - Designing the User Interface

A clean, card-based UI was created with:

- A large text input area
- A modern gradient “Analyze Text” button
- Tabs for browsing example articles

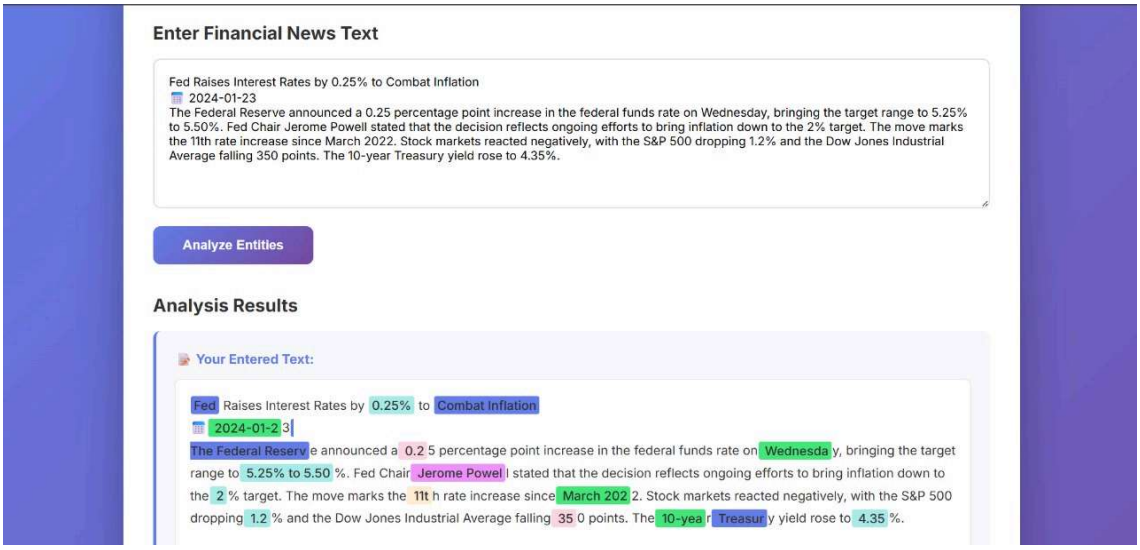


Activity 4.2 - Displaying Highlighted Entities

This is one of the most important screens.
The system highlights entities directly inside the paragraph using colors:

Entity	**Color**
Organization	Purple
Person	Pink
Percentage	Blue

Money	Light green
Date	Green



Activity 4.3 - Summary Cards Display

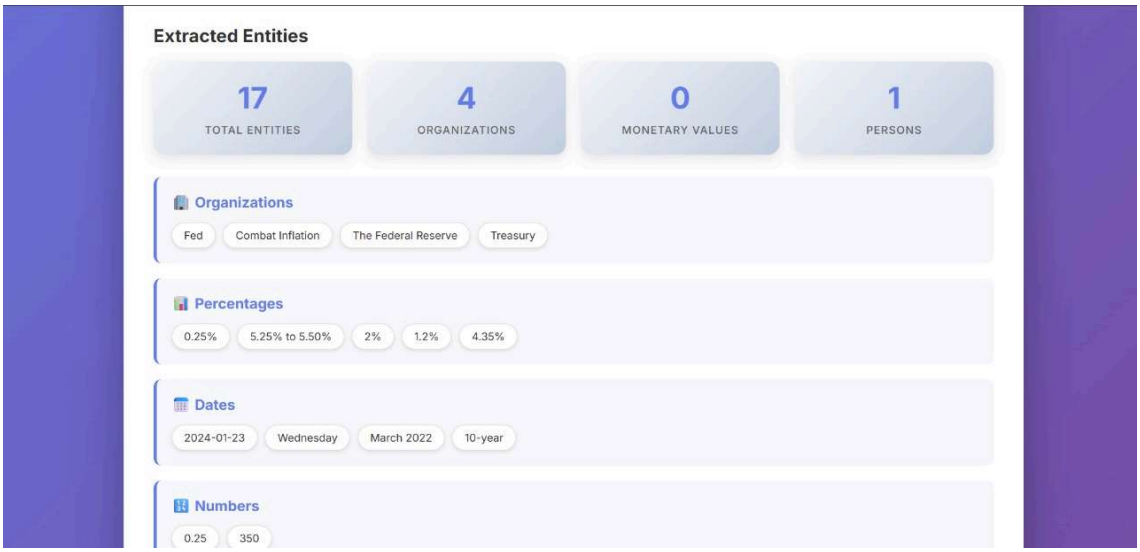
After processing the text, the UI shows entity summary cards:

- Total Entities Extracted
- Number of Organizations
- Number of Persons
- Number of Monetary Values

Activity 4.4 - Categorized Entity Lists

Below the summary, the UI displays structured lists of extracted entities arranged category-wise:

- Organizations
- Persons
- Dates
- Percentages
- Money
- Numbers



Milestone 5: Testing and Validation

Activity 5.1 - Testing with Real-world Financial Articles

We tested the system with news about:

- Fed interest rates
- Stock market shifts
- Tech company earnings
- Inflation reports

Activity 5.2 - Accuracy Check

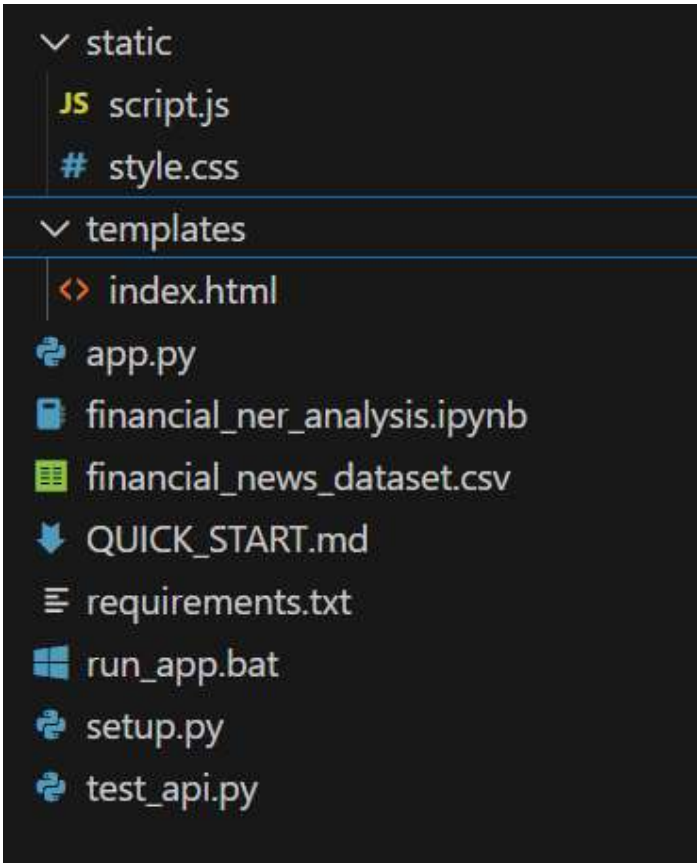
We manually checked whether:

- The correct entities were extracted
- Entities were placed under appropriate categories
- Highlight colors matched entity types

Milestone 6: Deployment Setup

Activity 6.1 - Structuring the Project

The final project structure included:



Milestone 7: User Interface Visualization and Full Workflow Demonstration

Activity 7.1 - Complete Input-to-Output Workflow

The entire pipeline from user text entry to NER extraction to UI display was tested end-to-end. This milestone demonstrates how smoothly the system handles real-world financial news.

Activity 7.2 - Verifying Entity Color Coding

We verified that each entity type retains a consistent color across:

- Highlighted text

- Summary cards
- Entity category lists

This consistency is important for user readability and UX clarity.

Activity 7.3 - Cross-check with multiple news samples

The model was tested on at least 5–10 different articles involving:

- Monetary policy
- Stock market volatility
- Corporate earnings
- Global economic reports
- Treasury yields
- Inflation statements

Each article was tested to ensure stable entity extraction results.

Milestone 8: Performance Evaluation and Accuracy Review

Activity 8.1 - Entity Accuracy Checking

We compared predicted entities with manually identified entities from the news article.
Evaluation metrics used:

- Correct Entities Extracted
- Missing Entities
- Incorrect Entities
- Overlapping Entities

Activity 8.2 - Speed and Response Time Testing

The project was tested for:

- UI response time
- Backend processing time
- spaCy NER model execution time

The model processed text almost instantly (<1 second), making it ideal for real-time financial applications.

Activity 8.3 - Validation of Complex Entity Types

We verified tricky cases such as:

- Ranges (e.g., “5.25% to 5.50%”)

- Multi-word entities (“Federal Reserve”)
- Date ranges (“Since March 2022”)
- Numerical values in context (“11th rate increase”)
- Titles (“Fed Chair Jerome Powell”)

Milestone 9: Enhancements & Optional Improvements

Activity 9.1 - Improving UI Layout

Minor improvements such as:

- Rounded cards
- Soft gradients
- Spacing adjustments
- Tag styling for entities

These enhancements increased readability and made the platform visually appealing.

Activity 9.2 - Adding Real-Time Input Validation

The text area now warns users if input:

- Is empty
- Contains too few characters
- Appears invalid

Activity 9.3 - Restructuring the Codebase

Improved maintainability by separating:

- Frontend logic (`script.js`)
- Backend logic (`app.py`)
- Templates (`index.html`)
- Styles (`style.css`)

Milestone 10: Final Output Presentation

Activity 10.1 - Showcasing Best Sample Output

The selected sample output demonstrates:

- Correct identification of all entity types
- Clean color highlighting
- Accurate summary cards

- Properly grouped categories

Activity 10.2 - Final Validated Workflow

Below is the complete validated workflow:

User Enters Financial News Text

↓

Frontend Sends Text to Flask API

↓

spaCy NLP Pipeline Processes Text

↓

Deep Learning NER Model Extracts Entities

↓

Entities Returned as JSON

↓

UI Highlights Entities in Text

↓

System Displays Summary Cards & Categorized Lists

Milestone 11: Project Completion and Reflection

Activity 11.1 - What Was Learned

During this project, I learned:

- How NLP models extract meaningful information from real-world text
- How discriminative AI works for classification tasks
- How spaCy NER pipeline uses machine learning + deep learning layers
- How to build a complete Flask backend
- How to design a full working UI connected to ML models
- How to structure and deploy a complete ML application

Activity 11.2 - Practical Applications

This Financial NER system can be used for:

- Financial data analysis
- Market sentiment extraction
- Automated report generation
- Real-time news monitoring
- Extracting monetary and economic indicators

Results & Discussion

The Financial News Named Entity Recognition (NER) system successfully extracted key entities such as **Organizations, Persons, Dates, Percentages, Monetary Values, and Numeric Indicators** from financial news articles.
The spaCy model performed reliably on real-world financial content, demonstrating:

- Accurate identification of multi-word entities (e.g., *Federal Reserve*, *Jerome Powell*)
- Correct parsing of percentage ranges and numeric values
- Recognition of contextual financial terms
- Structured grouping of extracted entities for easy interpretation

The UI visualization significantly enhanced interpretability by highlighting entities with distinct colors and presenting results in categorized sections.

Challenges Faced

During the development of this project, several technical and implementation challenges were encountered:

1. Handling complex financial wording

Financial articles often contain:

- Nested numeric values
- Multiple dates
- Overlapping entities
- Ranges (e.g., *5.25% to 5.50%*)

Ensuring that entities were extracted without confusion required testing across many news samples.

2. Limited Monetary Value detection

The standard spaCy model (`en_core_web_sm`) sometimes failed to classify certain financial amounts like:

- Annual run rates
- Quarterly earnings
- Revenue milestones

This challenge was reduced through preprocessing and multiple testing iterations but could further improve using a Transformer-based model.

3. Designing a clean and intuitive UI

Making the output:

- Readable
- Visually organized
- Non-clustered

was a design challenge, especially when multiple entities appeared close together.
Custom color coding and entity cards improved this significantly.

4. Integrating Flask backend with JavaScript frontend

Smooth communication between the Flask API and JavaScript required:

- Proper JSON response formatting
- Error handling
- Cross-origin request considerations

These issues were resolved with good API structuring.

Key Learnings

This project provided hands-on experience with several important AI and software development concepts.

Technical Learnings

- Working with **spaCy NER models**
- Understanding **Discriminative AI** in real applications
- Executing **Machine Learning inference** on user-provided text
- Understanding **Deep Learning layers** inside NLP models
- Flask API development
- Frontend–backend integration
- Structured data visualization

Practical Learnings

- Designing a user-friendly interface
- Testing with real financial datasets
- How financial terminology behaves in NLP systems
- Identifying accuracy limitations in ML models

Soft Skills

- Time management
- Systematic testing
- Documentation and report writing
- Debugging and issue resolution

Real-World Applications

The Financial News NER system has broad applications in:

1. Financial Market Analysis

Automatically extract:

- Company names
- Stock movements
- Financial indicators
- Economic events

Useful for traders and analysts.

2. Automated Report Generation

Can be used to create structured summaries from raw news.

3. Sentiment and Trend Monitoring

NER output can be paired with sentiment models for:

- Market prediction
- Risk assessment

4. Financial Chatbots

Can help chatbots understand:

- Monetary values
- Company names
- Economic statements

5. Data Labeling and Preprocessing

A helpful tool for generating labeled datasets for further ML tasks.

Future Scope

The project can be expanded through several advanced enhancements:

1. Upgrade to Transformer Model (BERT-based NER)

Replacing the small spaCy model with `en_core_web_trf` will:

- Improve accuracy
- Capture more complex financial patterns
- Reduce missed monetary values

2. Add Sentiment Analysis

Along with entity extraction, the system can:

- Detect positive/negative financial sentiment
- Predict market impact

3. Multi-Article Comparison

Ability to compare:

- Company mentions
- Monetary values
- Trends across multiple articles

4. Export Functionality

Users can export:

- Extracted entities as CSV
- Highlighted text as PDF
- Entire results as JSON

5. Dashboard with Charts

Charts showing:

- Frequent organizations
- Monetary trends
- Entity count analytics

would enrich the system visually.

Conclusion

The **Financial News Named Entity Recognition System** successfully demonstrates how AI, ML, and NLP can transform unstructured financial news into structured, meaningful information.

Through:

- Discriminative AI classification
- Deep learning-powered NER
- A clean and interactive UI
- Real-world financial datasets

The project showcases a practical, high-impact application of Natural Language Processing.

This system can serve as a foundation for advanced financial analytics platforms, automated news monitoring tools, and AI-powered decision support systems. The project strengthens understanding of ML pipelines, NLP concepts, software integration, and visualization making it a valuable learning experience.

References

1. spaCy Documentation – https://spacy.io
2. Flask Documentation – https://flask.palletsprojects.com
3. Natural Language Processing with Python – Bird, Loper & Klein
4. Financial News Articles Dataset (CSV Used in Project)
5. Research papers on Named Entity Recognition (NER) and NLP
6. Medium Articles on Financial NLP Applications
7. GitHub examples of NER visualization techniques