RPA Nano

In today's rapidly evolving technological landscape, the intersection of Robotic Process Automation (RPA) and artificial intelligence (AI) has ushered in a new era of efficiency and innovation. My journey into developing RPA nano services stems from a profound belief in leveraging technology to streamline processes traditionally reliant on manual programming.

What is RPA Nano?

Introduction

In today's rapidly evolving technological landscape, the intersection of Robotic Process Automation (RPA) and artificial intelligence (AI) has ushered in a new era of efficiency and innovation. My journey into developing RPA nano services stems from a profound belief in leveraging technology to streamline processes traditionally reliant on manual programming.

Background

Having immersed myself in the realm of software development, I observed firsthand the complexities and time-intensive nature of traditional programming tasks. Recognizing the potential of RPA to automate repetitive and rule-based activities, I embarked on a path to harness its transformative power. RPA nano services emerged as a focal point—an agile, modular approach designed to automate specific tasks with precision and scalability.

Development of RPA Nano Services

The development of RPA nano services revolves around dissecting complex programming tasks into granular components. Each nano service is meticulously crafted to address a distinct function, ranging from data extraction and validation to integration with existing systems. By encapsulating these functionalities into self-contained units, RPA nano services not only enhance operational efficiency but also reduce dependency on traditional programming paradigms.

Impact on Programming Paradigms

Central to my endeavor is the aspiration to redefine the role of programmers through RPA AI technology. By delegating routine programming tasks to automated processes, programmers are liberated to focus on higher-value activities such as algorithm design, system architecture, and

strategic innovation. This shift not only accelerates development cycles but also empowers teams to tackle more ambitious projects with agility and creativity.

Advantages of RPA AI Technology

The integration of AI within RPA further amplifies its capabilities, enabling intelligent decision-making and adaptive learning. Machine learning algorithms embedded within RPA systems continuously optimize processes, learning from data patterns to deliver increasingly refined outcomes. This symbiotic relationship between RPA and AI not only augments operational efficiency but also fosters a culture of continuous improvement within organizations.

Case Studies and Success Stories

Illustrating the impact of RPA nano services, numerous case studies exemplify their transformative influence across diverse industries. From automating financial reconciliations and customer service operations to enhancing supply chain management, organizations have realized significant cost savings, error reduction, and enhanced scalability through the adoption of RPA AI technologies.

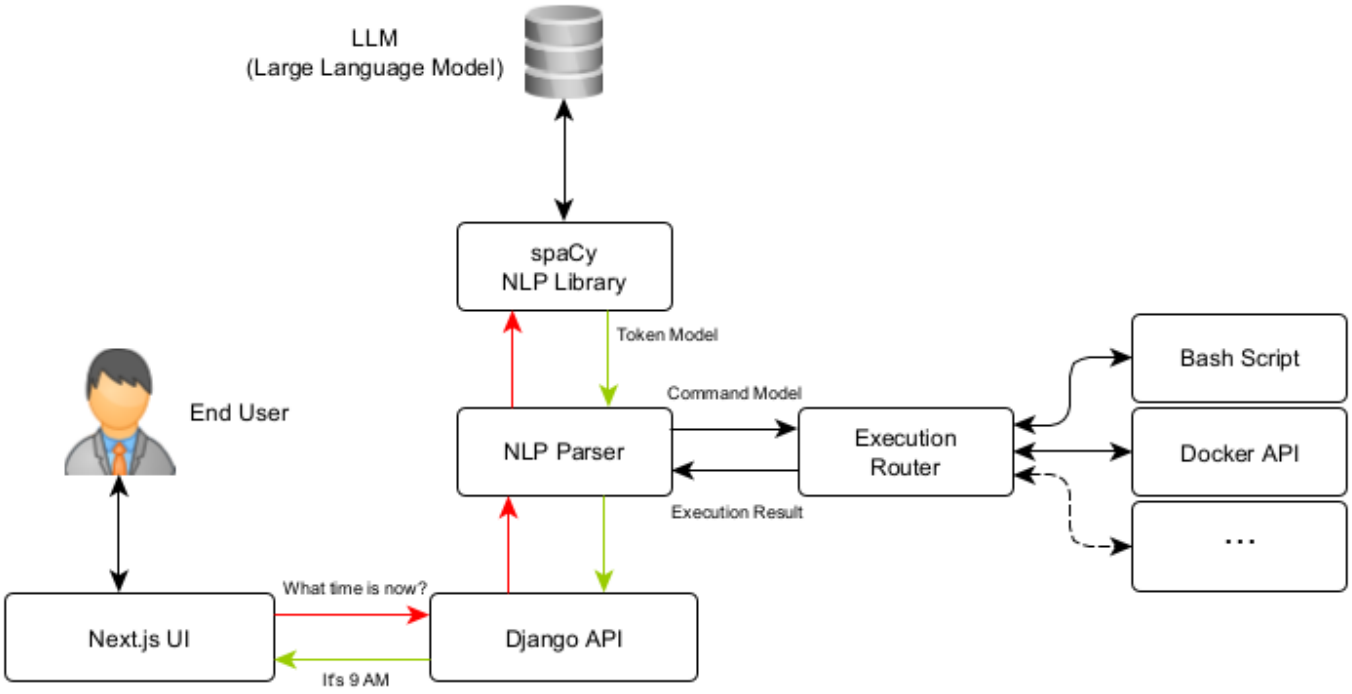
Future Outlook

Looking ahead, the trajectory of RPA nano services is poised for exponential growth. As organizations embrace digital transformation initiatives, the demand for agile, scalable automation solutions will continue to soar. By championing the convergence of RPA and AI, I envision a future where programming is redefined—not as a laborious task but as a dynamic collaboration between human ingenuity and machine intelligence.

Conclusion

In conclusion, my journey into developing RPA nano services represents a steadfast commitment to revolutionizing programming paradigms through innovation and automation. By harnessing the power of RPA AI technology, we embark on a transformative journey—one where efficiency, scalability, and creativity converge to shape the future of technology-driven enterprises.

Workflow



1. End user inputs text from UI
2. Frontend sends that input via API
3. API calls NLP Parser
4. NLP Parser calls LLM
5. NLP Parser gets token model from spaCy Library
6. NLP process the token model and results command model
7. Execution Router process the command model, maps and sends it to Execution Library
8. Execution Library returns execution result and Execution Router sends it to NLP Parser
9. NLP Parser packages the result into human readable format and sends it to API
10. API sends that result to frontend
11. Frontend shows the result into end user

API Server Installation

Preparation

```
apt install python3-pip
apt install python3.12-venv
python3 -m venv env
source env/bin/activate
```

Step 1 - Git clone

Clone the repository from Github:

```
git clone https://github.com/ahmadsidrap/rpa-nano.git
```

Step 2 - Install libraries

```
cd rpaNanoRouter
pip install -r requirements.txt
```

Step 3 - Migrate DB

```
python manage.py migrate
```

Step 4 - Create Superuser

```
python manage.py createsuperuser
```

Step 5 - Run the server

```
python manage.py runserver
```

Run RPA Router as Service

Create service file

```
sudo nano /etc/systemd/system/rpa-nano.service
```

```
[Unit]
Description=RPA Nano
After=network.target

[Service]
User=root
WorkingDirectory=/root/docker/rpa-nano/rpaNanoRouter
ExecStart=/root/docker/rpa-nano/rpaNanoRouter/env/bin/python3.12 /root/docker/rpa-nano/rpaNanoRouter/manage.py runserver 0.0.0.0:3002
Restart=always

[Install]
WantedBy=multi-user.target
```

Activate service

```
sudo systemctl daemon-reload
sudo systemctl start rpa-nano
sudo systemctl enable rpa-nano
```

Verify the status:

```
sudo systemctl status rpa-nano
```

Run Server with Virtual Environment

Installation

```
sudo apt install python3-virtualenv -y
```

Build virtual environment.

```
virtualenv env
```

Activate virtual environment.

```
source env/bin/activate
```

NLP Parser

NLP Parser parses the text into token model. Token model is a model that contains the structure of the text. Parser classifies these model into several queries.

Query and Transformation

Query is a set pattern that will parse token model. The matches result will be transformed into command model. The command model properties are as following.

- `command` - The command that will tell **Execution Router**
- `target` - The target of execution
- `related_tokens` - The tokens that are related to the target. For example, in `current date`, `current` is related tokens
- `type` - The type of command
- `source` - It can be an input for the target for execution

1. WhatQuery

The match pattern:

1. Index 0 → Position: `DET`
2. Index 1 → Position: `NOUN`
3. Index 2 → Position: `AUX`, optional
4. Index 3 → Position: `ADV`
5. Index 4 → Position: `PUNCT`, optional

Example matches:

- What time is it now?
- What time is now?
- What time now?

Token transformation:

1. `command`: `NOUN`

2. WhoQuery

The match pattern:

1. Index 0 → Position: `PRON`
2. Index 1 → Position: `AUX`
3. Index 2 → Position: `PRON`

Example matches:

- Who are you?
- Who is that?

Token transformation:

None

3. AuxiliaryQuery

Example matches:

1. Index 0 → Position: `NOUN`
2. Index 1 → Position: `ADV`, optional
3. Index 2 → Position: `AUX`
4. Index 3 → Position: `PUNCT`, optional

Example matches:

- Current time is?

Token transformation:

1. `command`: `NOUN`

4. CommandQuery

Pattern 1

Example matches:

1. Index 0 → Position: `VERB`
2. Index 1 → Position: `PRON`, optional
3. Index 2 → Position: `DET`, optional
4. Index 3 → Position: `ADJ`, optional
5. Index 4 → Position: `NOUN`

Example matches:

- Show me directory media

Pattern 2

Example matches:

1. Index 0 → Position: `VERB`
2. Index 1 → Position: `PRON`, optional
3. Index 2 → Position: `DET`, optional
4. Index 3 → Position: `ADJ`, optional
5. Index 4 → Position: `PROPN`

Example matches:

- Show me running containers

Token transformation:

1. `target`: `NOUN` or `PROPN`

API References

POST /api/rpa/nlp

Usage:

```
curl -X POST -H "Content-Type: application/json" -d '{"message": "what time is it now?"}'
http://localhost:8000/api/rpa/nlp
```

NLP Library Installation

For English:

```
python -m spacy download en_core_web_sm
```