Vendor Management System Documentation

1.Introduction

Purpose
The Vendor Management System (VMS) is a Django-based web application designed to streamline vendor management processes. It allows users to manage vendors, purchase orders, and track vendor performance metrics

Scope
The scope of this document is to provide an overview of the VMS, including its features, installation instructions, usage guidelines, backend logic for performance metrics calculation, and additional considerations for system efficiency and data integrity

2. System Overview

Features

- CRUD operations for vendors and purchase orders
- Calculation and tracking of vendor performance metrics
- RESTful API for integration with other systems

Technologies Used

- Django: Python web framework for backend development
- Django REST Framework: Toolkit for building Web APIs in Django
- PostgreSQL: Relational database management system for data storage
- Other Python libraries and tools as required

3. Installation

Prerequisites
- Python
- PostgreSQL database server
- Django and other dependencies (install via pip)

Installation Steps

- Clone the repository from [GitHub URL].
- Create and activate a virtual environment.
- Install required dependencies using pip.
- Configure PostgreSQL database settings in settings.py.
- Run migrations to create database schema.
- Start the Django development server

4. Usage

API Endpoints

Vendor Endpoints**:**

- GET /api/vendors/: List all vendors
- GET /api/vendors/<int:vendor_id>/: Retrieve details of a specific vendor
- POST /api/vendors/: Create a new vendor
- PUT /api/vendors/<int:vendor_id>/: Update an existing vendor
- DELETE /api/vendors/<int:vendor_id>/: Delete a vendor
- GET /api/vendors/<int:vendor_id>/performance/: Retrieve performance metrics of a vendor

Purchase Order Endpoints:

- GET /api/purchase_orders/: List all purchase orders
- GET /api/purchase_orders/<int:po_id>/: Retrieve details of a specific purchase order
- POST /api/purchase_orders/: Create a new purchase order
- PUT /api/purchase_orders/<int:po_id>/: Update an existing purchase order
- DELETE /api/purchase_orders/<int:po_id>/: Delete a purchase order

Examples

Creating a vendor

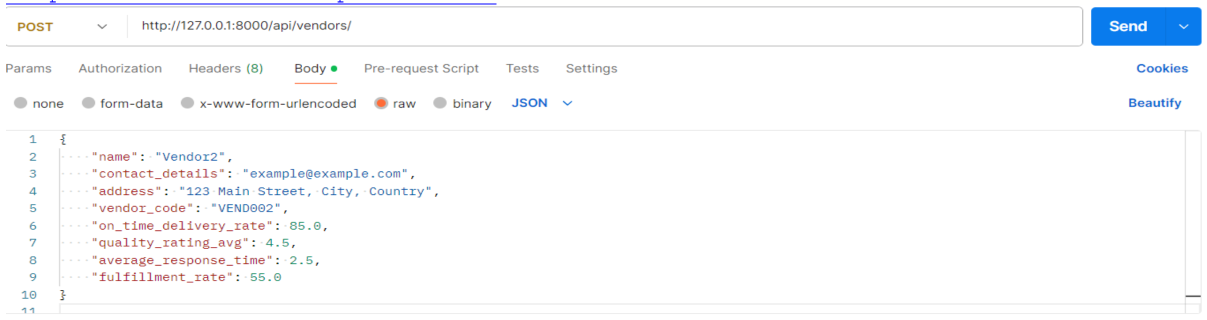
```
{
    "name": "Vendor2",
    "contact_details": "example@example.com",
    "address": "123 Main Street, City, Country",
    "vendor_code": "VEND002",
    "on_time_delivery_rate": 85.0,
    "quality_rating_avg": 4.5,
    "average_response_time": 2.5,
    "fulfillment_rate": 55.0
}
```

Creating a Purchase order

```
{
    "po_number": "PO02",
    "vendor": 2,
    "order_date": "2024-05-07T08:00:00Z",
    "delivery_date": "2024-05-08T01:00:00Z",
    "items": [
        {
            "name": "Product C",
            "quantity": 5,
            "unit_price": 30.00
        },
        {
            "name": "Product D",
            "quantity": 3,
            "unit_price": 25.00
        }
    ],
    "quantity": 8,
    "status": "completed",
    "quality_rating": 4.5,
    "issue_date": "2024-05-07T08:00:00Z"
}
```

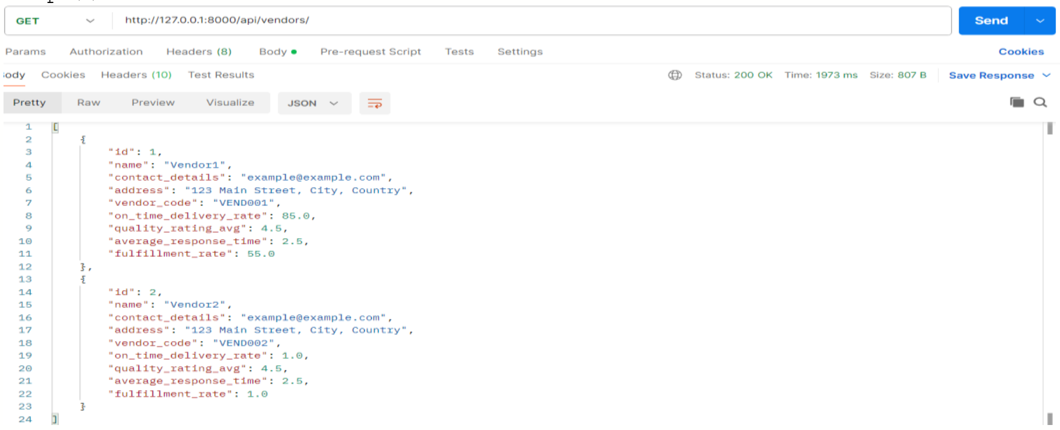
API list and Payloads Screenshots

http://127.0.0.1:8000/api/vendors/ -- Vendor API Post method



```
{
    "name": "Vendor2",
    "contact_details": "example@example.com",
    "address": "123 Main Street, City, Country",
    "vendor_code": "VEND002",
    "on_time_delivery_rate": 85.0,
    "quality_rating_avg": 4.5,
    "average_response_time": 2.5,
    "fulfillment_rate": 55.0
}
```

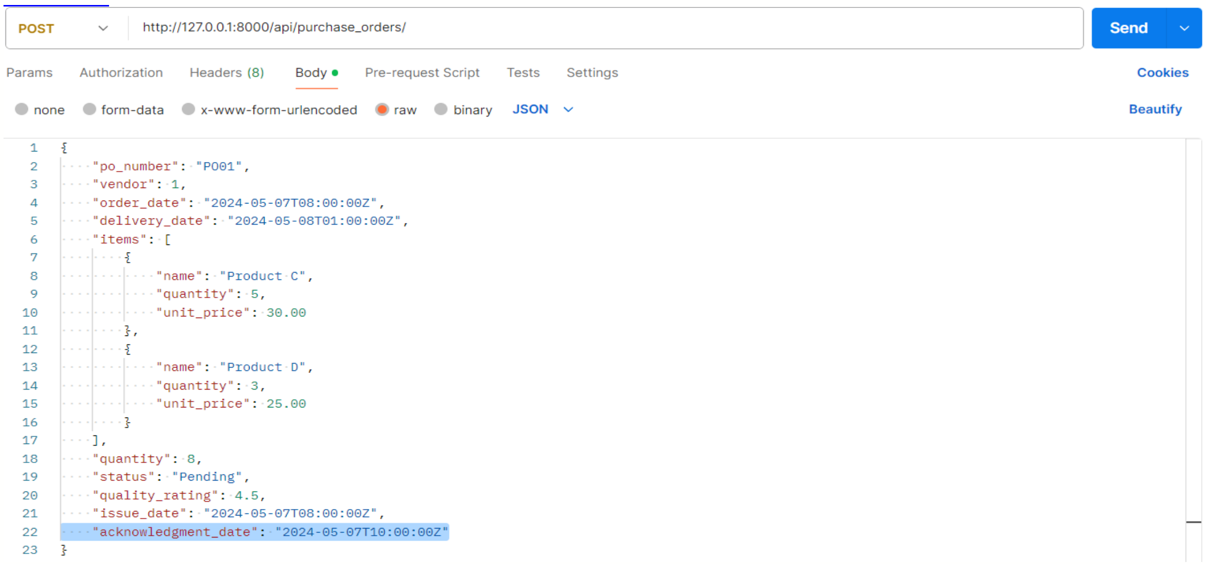
http://127.0.0.1:



```
[
    {
        "id": 1,
        "name": "Vendor1",
        "contact_details": "example@example.com",
        "address": "123 Main Street, City, Country",
        "vendor_code": "VEND001",
        "on_time_delivery_rate": 85.0,
        "quality_rating_avg": 4.5,
        "average_response_time": 2.5,
        "fulfillment_rate": 55.0
    },
    {
        "id": 2,
        "name": "Vendor2",
        "contact_details": "example@example.com",
        "address": "123 Main Street, City, Country",
        "vendor_code": "VEND002",
        "on_time_delivery_rate": 1.0,
        "quality_rating_avg": 4.5,
        "average_response_time": 2.5,
        "fulfillment_rate": 1.0
    }
]
```

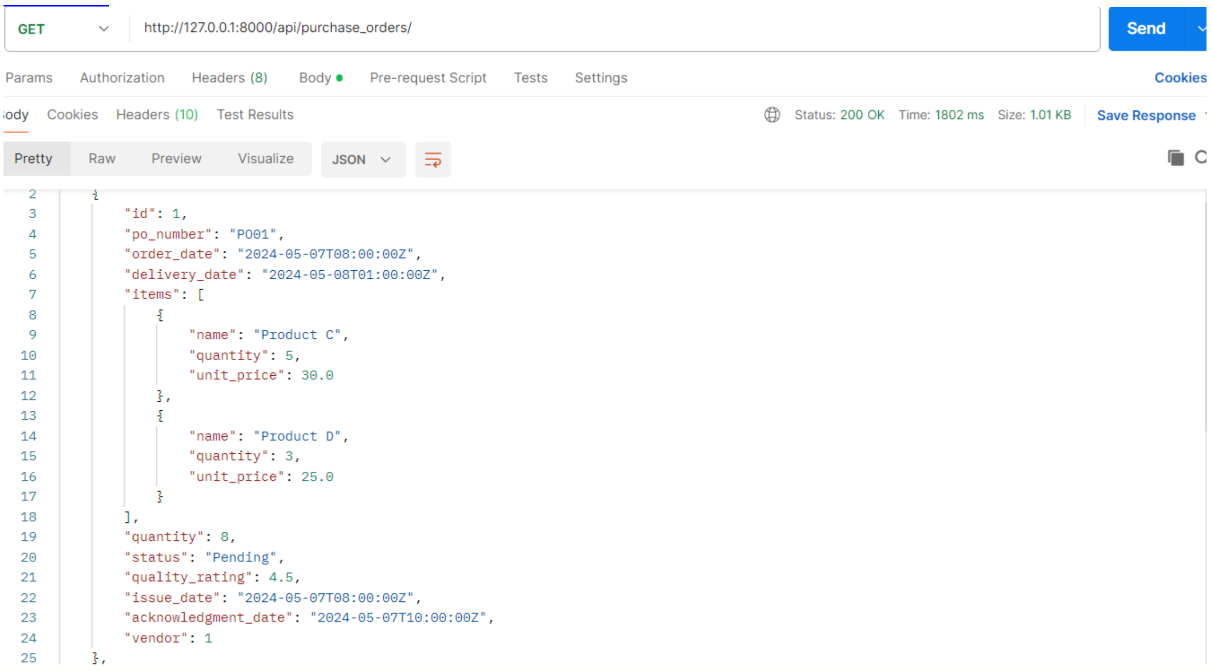
8000/api/vendors/ -- Vendor API Get method

http://127.0.0.1:8000/api/purchase_orders/ --Purchase order post method



```
{
    "po_number": "P001",
    "vendor": 1,
    "order_date": "2024-05-07T08:00:00Z",
    "delivery_date": "2024-05-08T01:00:00Z",
    "items": [
        {
            "name": "Product C",
            "quantity": 5,
            "unit_price": 30.00
        },
        {
            "name": "Product D",
            "quantity": 3,
            "unit_price": 25.00
        }
    ],
    "quantity": 8,
    "status": "Pending",
    "quality_rating": 4.5,
    "issue_date": "2024-05-07T08:00:00Z",
    "acknowledgment_date": "2024-05-07T10:00:00Z"
}
```

http://127.0.0.1:8000/api/purchase_orders/ --Purchase order Get method

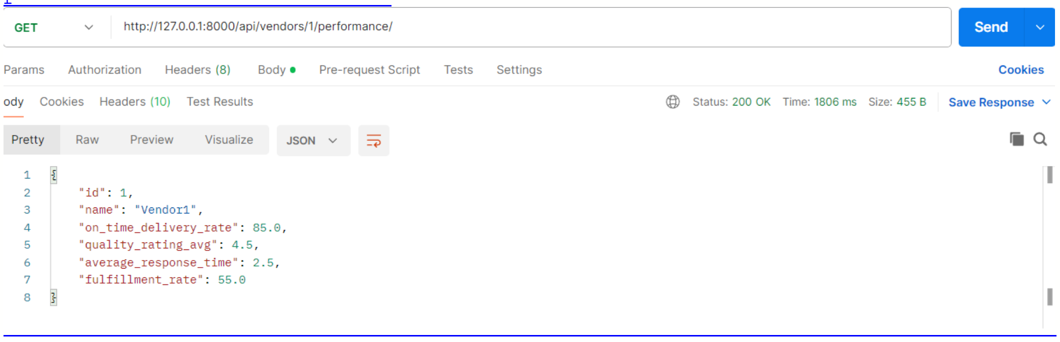


GET http://127.0.0.1:8000/api/purchase_orders/

Status: 200 OK Time: 1802 ms Size: 1.01 KB

```
    {
        "id": 1,
        "po_number": "P001",
        "order_date": "2024-05-07T08:00:00Z",
        "delivery_date": "2024-05-08T01:00:00Z",
        "items": [
            {
                "name": "Product C",
                "quantity": 5,
                "unit_price": 30.0
            },
            {
                "name": "Product D",
                "quantity": 3,
                "unit_price": 25.0
            }
        ],
        "quantity": 8,
        "status": "Pending",
        "quality_rating": 4.5,
        "issue_date": "2024-05-07T08:00:00Z",
        "acknowledgment_date": "2024-05-07T10:00:00Z",
        "vendor": 1
    },
```

http://127.0.0.1:8000/api/vendors/1/performance/ -Vendor performance Get method



GET http://127.0.0.1:8000/api/vendors/1/performance/

Status: 200 OK Time: 1806 ms Size: 455 B

```
{
    "id": 1,
    "name": "Vendor1",
    "on_time_delivery_rate": 85.0,
    "quality_rating_avg": 4.5,
    "average_response_time": 2.5,
    "fulfillment_rate": 55.0
}
```

5. Backend Logic

Performance Metrics Calculation

- **On-Time Delivery Rate**: Calculated upon completion of each PO with the 'completed' status.
- **Quality Rating Average**: Updated upon completion of each PO where a quality rating is provided.
- **Average Response Time**: Calculated upon acknowledgment of each PO by the vendor.
- **Fulfillment Rate**: Calculated upon any change in PO status.

Signals

- Signals are used to trigger metric updates in real-time when related PO data is modified.

6. Additional Considerations

Efficiency

- Backend logic for metric calculation is optimized for handling large datasets without significant performance issues.

Data Integrity

- Checks are included to handle scenarios like missing data points or division by zero in calculations to ensure data integrity.

Real-time Updates

- Django signals are used to trigger metric updates in real-time when related PO data is modified, ensuring real-time updates of metrics

7. Conclusion

- The Vendor Management System (VMS) is a valuable tool for businesses to manage vendors, track purchase orders, and monitor vendor performance metrics. Built using Django and Django REST Framework, the system offers scalability and efficiency.
- With optimized backend logic and real-time updates, the VMS ensures reliability and data integrity. Future improvements may include user authentication, advanced filtering, and enhanced error handling.
- In summary, the VMS streamlines vendor management processes, enhances efficiency, and empowers businesses with valuable insights into vendor performance.