Task Overview:

Objective: Develop a Yak Shop web shop that allows the Yak Shepherd to manage his herd, track stock, and fulfill customer orders. Additionally, implement AI/ML features for anomaly detection, behavior analysis, and provide recommendations for the Yak Shepherd.

Core Functionality:

1. Data Processing:

- ✓ Implement a program that reads a JSON file containing information about the herd
- ✓ Take two parameters: JSON file path and an integer T representing elapsed time in days.

Explanation: **Integer T representing elapsed time in days**:

This parameter signifies a numerical value (an integer) that represents a certain duration or elapsed time in days. This value will likely be used within the program's logic to calculate or simulate changes or events that occur within the herd after the specified time has passed.

2. Stock and Herd Management:

· Calculate and display the stock of milk and skins after T days.
· Provide a view of the herd after T days, including yak names, ages, and the last shaved age.

3. Order Fulfillment:

· Implement order fulfillment, checking stock availability and returning appropriate HTTP status codes and order details.

AI/ML Integration:

4. AI/ML Anomaly Detection:

· Develop anomaly detection models to identify unusual patterns in yak health or behavior, showcasing AI capabilities in monitoring herd well-being.

5. AI/ML Behavior Analysis:

· Create behavior analysis models to predict and understand yak behavior over time, offering insights into their habits and responses to different conditions.

6. AI/ML Recommendations:

· Provide recommendations for optimal herd management based on AI/ML analysis, such as adjusting feeding schedules or health interventions.

Gen AI / NLP Feature:

7. Gen AI / NLP Stock Management:
· Utilize NLP to update and check stock levels through natural language commands.
· Example: "Update milk stock by 100 liters" or "Check current skin inventory."

Data Generation:
· Create diverse datasets to simulate different scenarios, varying factors such as yak ages, health conditions, and behavior patterns.

Model Training:
· Clearly define and explain the AI/ML models used for anomaly detection, behavior analysis, and recommendations.
· Provide details on the training data sources, model architectures, and evaluation metrics.

Deployment and Documentation:
· Include instructions for deploying the YakShop webshop and integrating AI/ML models.
· Document the model deployment process, ensuring reproducibility.
· Develop comprehensive test cases for both core functionality and AI/ML features.

Communication:
· The candidate can reach out to us for any clarifications during the development process.

Evaluation Criteria:
· Successful implementation of core functionality.
· Effective integration of AI/ML features with clear explanations.
· Creativity and functionality in the Gen AI / NLP feature.
· Clarity and organization of code and documentation.
· Adequate test coverage and reproducible deployment process.

Sample JSON for Data Processing:

```
{
 "herd": [
   { "name": "Betty-1", "age": 4, "sex": "f" },
   { "name": "Betty-2", "age": 8, "sex": "f" },
   { "name": "Betty-3", "age": 9.5, "sex": "f" }
 ]
}
```

Sample Diverse Dataset:

```
{
 "herd": [
   { "name": "Betty-1", "age": 3, "health": "good", "behavior": "calm" },
   { "name": "Betty-2", "age": 7, "health": "excellent", "behavior": "playful" },
   { "name": "Betty-3", "age": 9.5, "health": "fair", "behavior": "stubborn" }
 ]
}
```

Sample Stock Data:

```
{
 "milk": 1104.480,
 "skins": 3
}
```

Additional Details:
· For the order fulfillment, you can create a sample order JSON, specifying the customer, order date (T), and the desired quantities of milk and skins.

Sample Order Data:

```
{
```

```
 "customer": "Medvedev",
 "order": {
   "milk": 1100,
   "skins": 3
 }
}
```


· Ensure to simulate different scenarios for order fulfillment, including cases where the order can be fully fulfilled, partially fulfilled, or not fulfilled at all.

· During AI/ML model training, use historical data from the herd, incorporating variations in age, health conditions, and behavior patterns.

These samples and additional details should help you cover the essential aspects of the assignment.

```
import json
import numpy as np
from sklearn.svm import SVC

class YakShop:
    def __init__(self, json_file_path, t):
        # Load the yak data from the JSON file
        with open(json_file_path, "r") as f:
            self.yak_data = json.load(f)

        # Set the elapsed time in days
        self.t = t

        # Initialize the AI/ML models
        self.anomaly_detection_model = SVC()
        self.behavior_analysis_model = SVC()
        self.recommendation_model = SVC()

        # Train the AI/ML models
        self.train_models()

    def calculate_stock(self):
        # Calculate the milk stock
        milk_stock = self.yak_data["milk_production_rate"] * self.t -
self.yak_data["milk_sales"]

        # Calculate the skin stock
        skin_stock = self.yak_data["skin_growth_rate"] * self.t -
self.yak_data["skin_sales"]

        return milk_stock, skin_stock

    def view_herd(self):
```

```
        # Get the yak names, ages, and last shaped ages
        yak_names = [yak["name"] for yak in self.yak_data]
        yak_ages = [yak["age"] for yak in self.yak_data]
        yak_last_shaped_ages = [yak["last_shaped_age"] for yak in self.yak_data]

        # Return the yak information
        return yak_names, yak_ages, yak_last_shaped_ages

    def fulfill_order(self, order):
        # Check the availability of stock
        milk_stock, skin_stock = self.calculate_stock()

        if order["milk_quantity"] > milk_stock or order["skin_quantity"] >
skin_stock:
            # Out of stock
            return 404, None
        else:
            # Order fulfilled
            return 200, order

    def detect_anomalies(self):
        # Get the yak health and behavior data
        yak_health_data = [yak["health_data"] for yak in self.yak_data]
        yak_behavior_data = [yak["behavior_data"] for yak in self.yak_data]

        # Make predictions with the anomaly detection model
        predictions =
self.anomaly_detection_model.predict(np.concatenate((yak_health_data,
yak_behavior_data), axis=1))

        # Return the yak names that have anomalies detected
        yak_names_with_anomalies = []
        for i in range(len(predictions)):
            if predictions[i] == 1:
                yak_names_with_anomalies.append(self.yak_data[i]["name"])

        return yak_names_with_anomalies

    def analyze_behavior(self):
```

```
        # Get the yak behavior data
        yak_behavior_data = [yak["behavior_data"] for yak in self.yak_data]

        # Make predictions with the behavior analysis model
        predictions = self.behavior_analysis_model.predict(yak_behavior_data)

        # Return the yak names and their predicted behaviors
        yak_names_and_behaviors = []
        for i in range(len(predictions)):
            yak_names_and_behaviors.append((self.yak_data[i]["name"],
predictions[i]))

        return yak_names_and_behaviors

    def recommend_actions(self):
        # Get the yak health and behavior data
        yak_health_data = [yak["health_data"] for yak in self.yak_data]
        yak_behavior_data = [yak["behavior_data"] for yak in self.yak_data]

        # Make predictions with the recommendation model
        predictions =
self.recommendation_model.predict(np.concatenate((yak_health_data,
yak_behavior_data), axis=1))

        # Return the yak names and their recommended actions
        yak_names_and_recommended_actions = []
        for i in range(len(predictions)):

yak_names_and_recommended_actions.append((self.yak_data[i]["name"],
predictions[i]))

        return yak_names_and_recommended_actions

    def train_models(self):
        # Load the training data
        with open
```