**Problem Definition:**

UPS deliveries are frequently left at customers' homes, where they can be exposed to rain, resulting in damaged goods. To mitigate these occurrences, we aim to implement an automated system that analyzes weather conditions. If there is a forecast of rain or storms, the system will either postpone the delivery, suggest the customer pick up the package, or request their preference for delivery.

**IFTTT:**

* <https://ifttt.com/docs/process_overview>
* <https://ifttt.com/docs/api_reference>
* Endpoint tests : <https://ifttt.com/services/delivery_track/api/endpoint_tests>
* Testing documentation : <https://ifttt.com/docs/testing#example>
* Service API requirements: <https://ifttt.com/docs/api_reference>
* IFTTT documentations are some of the worst I have seen. Also the video tutorials are incomplete.
* When using AMBIENT WEATHER it is necessary to own a supported device. We can connect the device using it’s MAC addresses. (Reference : <https://ambientweather.com/faqs/question/view/id/1450/> )
* Same goes with Netatmo weather station and also Bloomsky weather.( I have not been able to connect with those yet, some kind of signup authentication error occurs. Maybe not supported in BD)

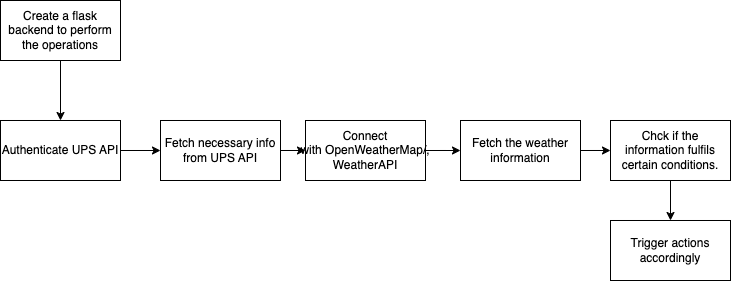
**Apple Shortcuts:**

* Created a shortcut that retrieves current weather
* If current weather is above 25 degree celsius, display a text. Otherwise do nothing.

But, I think these are not suitable for our use cases. Because we need to have a station at every location we need to track. Because these rules are set for each station.

However we can go for a simpler approach if it works.

**Workflow:**



* First, we need to create a backend for acting as a middleware between the services. Flask would be better for this approach.
* Authenticate UPS API and fetch the information like - order pickup location, delivery location, weight etc.
* Connect with OpenWeathrMap/ WeatherAPI to fetch the weather update at the delivery location.
* Check if the weather meets the conditions.
* Trigger actions accordingly.

**N.B :** Middleware service: a middleware service refers to a server-side application that acts as an intermediary layer between different services and components in your overall system architecture. It facilitates communication, data processing, and integration between these components, ensuring they work together seamlessly.

Task to do:

1. Follow this process to set up an applet <https://chatgpt.com/share/c43c713e-bb8c-4692-bdc1-9ed7d4307f1c>

Zapier- Shopify order status: <https://zapier.com/apps/edgility/integrations/shopify>

Creating applet in IFTTT for order status tracking :

#### **1. Set Up Your Local Server**

Let's set up a Flask server that checks the order status using the UPS API and triggers the appropriate IFTTT webhook if a delivery is late.

**Install Required Libraries**:  
bash  
Copy code  
pip install Flask requests

**Create the Flask App**:  
python  
Copy code  
from flask import Flask, request, jsonify

from datetime import datetime

import requests

app = Flask(\_\_name\_\_)

# UPS API credentials (example placeholders)

UPS\_API\_KEY = 'YOUR\_UPS\_API\_KEY'

UPS\_USERNAME = 'YOUR\_UPS\_USERNAME'

UPS\_PASSWORD = 'YOUR\_UPS\_PASSWORD'

# Demo order list

orders = {

"1": {"tracking\_number": "1Z12345E0291980793", "expected\_delivery\_date": "2023-06-25", "email": "user@example.com", "phone": "+1234567890", "preference": "email"},

"2": {"tracking\_number": "1Z12345E0291980794", "expected\_delivery\_date": "2023-06-27", "email": "user2@example.com", "phone": "+0987654321", "preference": "sms"}

}

def fetch\_order\_status(tracking\_number):

# Simulated response for demonstration

return {"status": "delayed"}

def check\_and\_notify\_orders():

current\_date = datetime.now().date()

for order\_id, order\_info in orders.items():

tracking\_number = order\_info["tracking\_number"]

expected\_delivery\_date = datetime.strptime(order\_info["expected\_delivery\_date"], "%Y-%m-%d").date()

order\_status = fetch\_order\_status(tracking\_number)

if order\_status and order\_status["status"] == "delayed":

if current\_date > expected\_delivery\_date:

if order\_info["preference"] == "email":

trigger\_ifttt\_webhook("delivery\_late\_email", order\_info["email"], order\_id)

elif order\_info["preference"] == "sms":

trigger\_ifttt\_webhook("delivery\_late\_sms", order\_info["phone"], order\_id)

def trigger\_ifttt\_webhook(event\_name, value1, value2):

ifttt\_event\_url = f'https://maker.ifttt.com/trigger/{event\_name}/with/key/YOUR\_IFTTT\_KEY'

payload = {"value1": value1, "value2": value2}

requests.post(ifttt\_event\_url, json=payload)

@app.route('/check-orders', methods=['POST'])

def check\_orders():

check\_and\_notify\_orders()

return jsonify({"status": "Order checks complete"}), 200

if \_\_name\_\_ == '\_\_main\_\_':

app.run(port=5000)

#### **2. Create IFTTT Applets**

**Email Notification Applet**:

1. Go to [IFTTT](https://ifttt.com/).
2. Click on "Create".
3. Click on "If This".
4. Search for "Webhooks" and select it.
5. Choose "Receive a web request" and set the event name to delivery\_late\_email.
6. Click on "Create Trigger".
7. Click on "Then That".
8. Search for "Email" and select "Send me an email".
9. Configure the email content using {{Value1}} and {{Value2}} (e.g., "Your order {{Value2}} is delayed.").
10. Click on "Create Action".

**SMS Notification Applet**:

1. Go to [IFTTT](https://ifttt.com/).
2. Click on "Create".
3. Click on "If This".
4. Search for "Webhooks" and select it.
5. Choose "Receive a web request" and set the event name to delivery\_late\_sms.
6. Click on "Create Trigger".
7. Click on "Then That".
8. Search for "SMS" and select "Send me an SMS".
9. Configure the SMS content using {{Value1}} and {{Value2}} (e.g., "Your order {{Value2}} is delayed.").
10. Click on "Create Action".

#### **3. Schedule Periodic Checks**

Use a scheduler to periodically call the /check-orders endpoint of your Flask server. This can be done using a simple script or a task scheduler like cron.

**Scheduler Script**:

Create a script that sends a POST request to the Flask server at regular intervals:  
python  
import requests

import time

while True:

response = requests.post('http://localhost:5000/check-orders')

print(response.json())

# Wait for a specified interval (e.g., 24 hours)

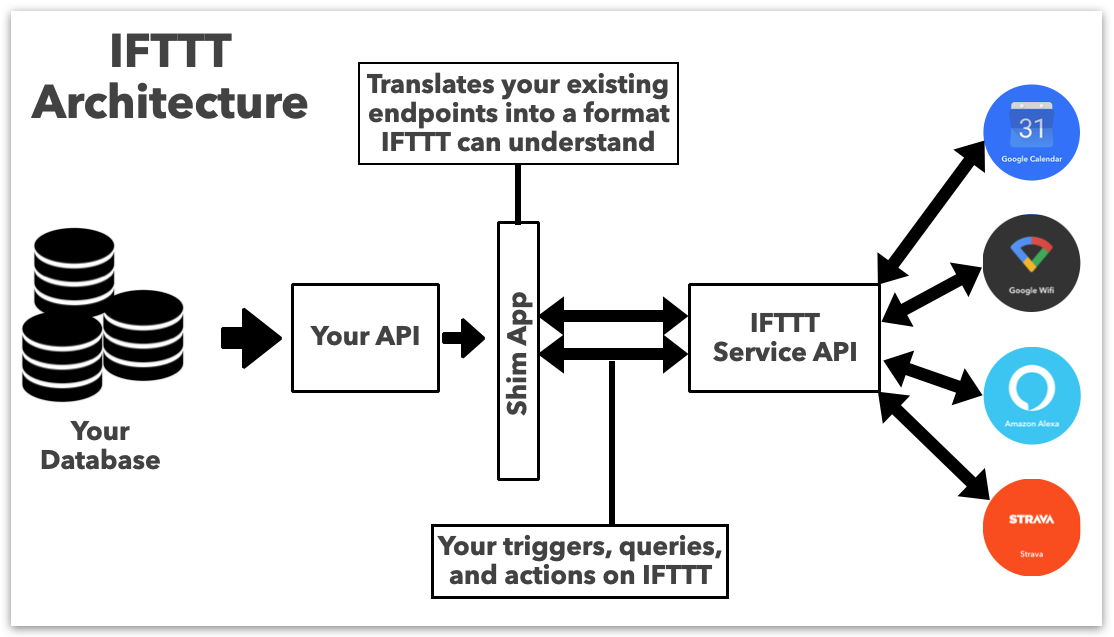
time.sleep(86400) # 24 hours in seconds

### **Summary**

* **Flask Server**: Checks the order status using the UPS API and triggers IFTTT webhooks if a delivery is late.
* **IFTTT Applets**: Configured to send notifications via email or SMS based on the triggered webhook.
* **Scheduler Script**: Periodically triggers the order status check.

This setup allows users to customize their notification preferences via IFTTT, while your local server handles the logic for detecting late deliveries and triggering notifications.

<https://chatgpt.com/share/98beb707-4696-4d5f-adcd-f987315fc69f>



“Shim app” is the middleware we mentioned previously.

**Zapier**

[**https://zapier.com/**](https://zapier.com/)

How its different from user facing platforms such as IFTTT and Apple shortcuts?

How Zapier provide automation service to different website/companies

What is Zap?

Zapier private integration : <https://platform.zapier.com/quickstart/build-private-integration>

Zapier cli vs zapier UI : <https://platform.zapier.com/quickstart/ui-vs-cli>

Creating integration in Zapier : <https://platform.zapier.com/quickstart/ui-tutorial>

Trigger zaps from webhooks : <https://help.zapier.com/hc/en-us/articles/8496288690317-Trigger-Zaps-from-webhooks>

Zapier not receiving request properly : <https://help.zapier.com/hc/en-us/articles/8496215655437-Zap-isn-t-receiving-webhooks>

Zapier Chatbots: <https://zapier.com/ai/chatbot>

Question: okay, if i decided to make a service in zapier, where users can integrate my service in actions/triggers in zaps. I mean, i want the users to use the apis and create their own triggers/actions. I hope I can make you understand. Then what should be my approach?

Answer :

<https://chatgpt.com/share/00b734a3-c39c-4dbf-9868-4950706d1fd9>

Creating custom integration in zapier : <https://www.youtube.com/playlist?list=PLBS1L3Ug2VVrZShhy4tfq-urDoK1rcx6Z>