Custom OCR system

Business Cards OCR System

Business Cards OCR

Loom - what it’s all about

Enhancing Communication with Business Cards

Presentation

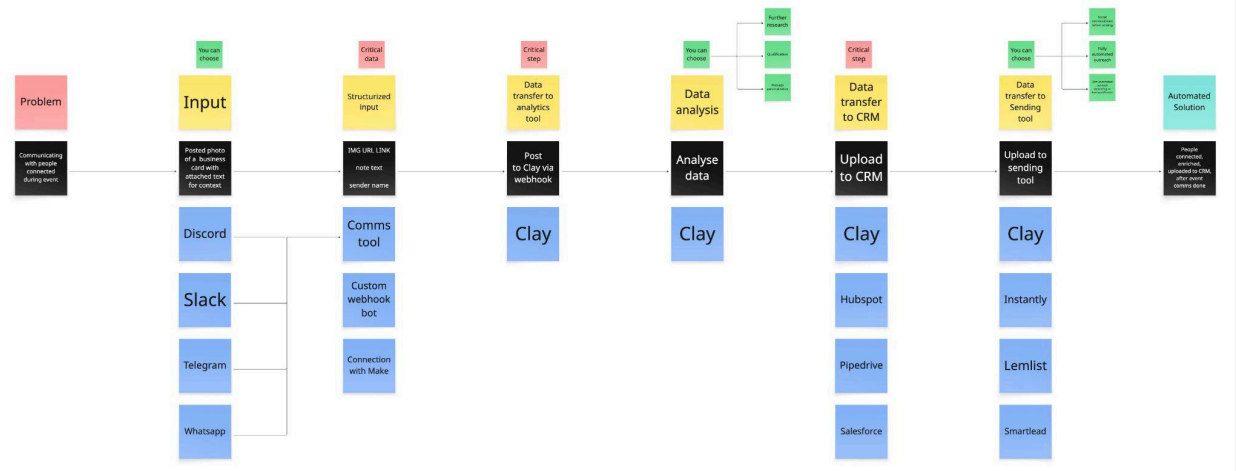
Business Cards OCR - Clay presentation

Clay Table - core framework

▶️https://app.clay.com/shared-table/share_0sxlwq86gwHafJzssPt ◀️

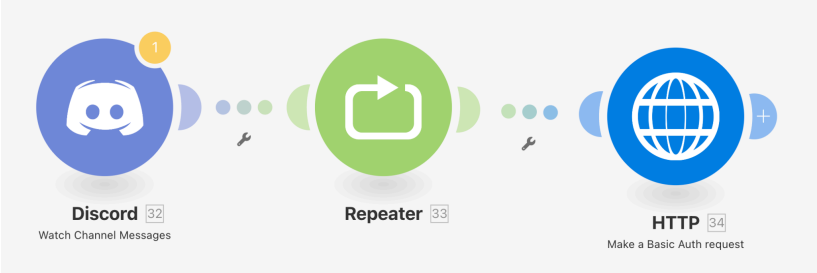
Framework chart

Business Card OCR - chart.pdf ⏪ better quality



Make.com easiest scenario

1. Create connection Make x Discord.
    a. How to create a proper connection Discord x Make
2. Start with trigger “Watch channel messages”.
3. Set repeater:
    a. Initial value: 1
    b. Repeats: 1
4. Use HTTP “Make a Basic Auth request”.
5. Post data to CLAY webhook.



Warning: automation will work every 15-minutes analysing messages on Discord. It will work but with delay.

Other option: it is possible to automate the whole process in Make but with huge data analysis limitations in comparison to CLAY.

Best option: Use Discord Webhook POSTing BOT for instant delivery.

Discord Webhook BOT (python) - instant trigger

Role

Every time there is a message sent in any channel on Discord send all information to webhook.

BOT - code

```
import discord
import requests

intents = discord.Intents.default()
intents.message_content = True

client = discord.Client(intents=intents)

WEBHOOK_URL = "YOUR_WEBHOOK_URL"

@client.event
async def on_ready():
    print(f'Bot zalogowany jako {client.user}')

@client.event
async def on_message(message):
    if message.author.bot:
        return

    img_url = None
    proxy_img_url = None

    if message.attachments:
        first_attachment = message.attachments[0]
        img_url = first_attachment.url
        proxy_img_url = first_attachment.proxy_url

    data = {
        "author_name": f"{message.author.name}#{message.author.discriminator}",
        "author_id": str(message.author.id),
        "content": message.content,
        "date": str(message.created_at),
        "img_url": img_url,
        "proxy_img_url": proxy_img_url
    }
```

```
    requests.post(WEBHOOK_URL, json=data)

client.run("YOUR_DISCORD_KEY")
```

Webhook BOT Deployment

GOAL

The goal is to containerize and deploy it to Kubernetes so that every message posted on Discord is sent to a webhook.

Overview of the Steps

1. Prepare the project structure
2. Create `requirements.txt`
3. Write the `Dockerfile`
4. Build and test Docker locally
5. Push the Docker image (optional)
6. Deploy to Kubernetes (from scratch)
7. Create Kubernetes secrets
8. Sample deployment manifests

---

1. Project Structure

Organize your project as follows:

```
discord-bot/
├── bot.py
```

```
├── requirements.txt
├── Dockerfile
├── .dockerignore
├── k8s/
│   ├── deployment.yaml
│   └── service.yaml
```

---

2. `requirements.txt`

Contents:

txt

```
discord.py==2.3.2
requests
```

---

3. `Dockerfile`

Dockerfile

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["python", "bot.py"]
```

---

4. Build and Test Docker Locally

Update your `bot.py` to use environment variables:

python

```
import os
WEBHOOK_URL = os.environ["WEBHOOK_URL"]
DISCORD_KEY = os.environ["DISCORD_KEY"]
```

Replace your static values like `"YOUR_WEBHOOK_URL"` and `"YOUR_DISCORD_KEY"` accordingly.

Then build and run:

bash

```
docker build -t discord-bot .

docker run -e DISCORD_KEY=xxx -e WEBHOOK_URL=https://your.webhook
discord-bot
```

---

5. Kubernetes Deployment

Let's assume a Kubernetes namespace called `tools`.

`k8s/deployment.yaml`:

yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: discord-bot
  namespace: tools
spec:
  replicas: 1
  selector:
    matchLabels:
      app: discord-bot
  template:
    metadata:
      labels:
```

```
      app: discord-bot
  spec:
    containers:
      - name: bot
        image: registry.gitlab.com/YOUR_GROUP/discord-bot:latest
        env:
          - name: DISCORD_KEY
            valueFrom:
              secretKeyRef:
                name: discord-bot-secrets
                key: DISCORD_KEY
          - name: WEBHOOK_URL
            valueFrom:
              secretKeyRef:
                name: discord-bot-secrets
                key: WEBHOOK_URL
```

`k8s/service.yaml` (optional)

Only needed if you plan to expose metrics or debug info:

yaml

```
apiVersion: v1
kind: Service
metadata:
  name: discord-bot
  namespace: tools
spec:
  selector:
    app: discord-bot
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8000
```

Kubernetes Secret

Use the command below to create the secret holding your Discord bot token and webhook URL:

bash

```
kubectl create secret generic discord-bot-secrets \
  --from-literal=DISCORD_KEY='your_discord_bot_token' \
  --from-literal=WEBHOOK_URL='https://your.webhook.url' \
  -n tools
```

Push Docker Image to GitLab Registry (optional)

Sample `.gitlab-ci.yml`

yaml

```
image: docker:latest

services:
  - docker:dind

variables:
  DOCKER_DRIVER: overlay2

stages:
  - build

build:
  stage: build
  script:
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD"
$CI_REGISTRY
    - docker build -t $CI_REGISTRY_IMAGE:latest .
    - docker push $CI_REGISTRY_IMAGE:latest
```

Deploy to Kubernetes

Option A: Manual deployment

bash

```
kubectl apply -f k8s/deployment.yaml
```

Option B: GitLab CD job

You can add a `deploy` job to your pipeline to apply manifests automatically, using `kubectl` and a Kubeconfig file.

Final Checklist

- [ ] Bot uses `WEBHOOK_URL` and `DISCORD_KEY` from env vars
- [ ] Docker image works locally
- [ ] Docker image pushed to GitLab registry
- [ ] Kubernetes secrets created
- [ ] Deployment applies correctly
- [ ] Bot shows up online in Discord
- [ ] Messages are successfully POSTed to webhook

To prepare:

- [ ] `Dockerfile`,
- [ ] `deployment.yaml`,
- [ ] `.gitlab-ci.yml`)

Ask GPT for help if needed.