📧 Bulk Email Campaign Management System

Production-Ready Django + Celery + SMTP + IMAP Bounce Processing

This project is a fully functional bulk email campaign engine designed to create, schedule, send, and track email campaigns at scale. It includes recipient management, CSV upload, campaign scheduler, bounce tracking via IMAP, Celery-based async workers, and a MaterializeCSS UI.

🚀 Features

✅ Campaign Management

- Create campaigns with
  - Name
  - Subject
  - HTML / Text content
  - Scheduled time
  - Groups or All Recipients
- Status tracking (Draft → Scheduled → In Progress → Completed)
- Recipient targeting by group

✅ Recipient Management

- Upload recipients via CSV (name,email,subscription_status)
- Auto-validation (email format, duplicates)
- Create & assign groups
- Add to campaigns
- Edit / Delete recipients

✅ Email Sending Engine (Celery)

- Auto triggers at scheduled time
- Background email processing
- Scalable parallel workers
- Delivery logs for each email

✅ Bounce Processing (IMAP)

- Connects to Gmail IMAP
- Fetches **Delivery Status Notification (Failure)**
- Extracts campaign ID via [CID:x] in subject

- Updates failed logs & bounce table
- Shown in **Bounce Reports**

✅ **Dashboard**

- Total campaigns
- Recipients count
- Sent / Failed count
- Recent campaigns
- Auto Trigger button
- Status summary

✅ **Reports**

- Failed Report CSV
- Campaign report summary
- Auto-email report to admin

🏗️ **System Architecture**

UI (Django + MaterializeCSS)

|

Django Views → Models → Celery Tasks

|

Celery Worker + Celery Beat

|

SMTP Server (Gmail)

|

IMAP Bounce Processor

|

PostgreSQL DB

📁 **Project Structure**

core/

campaigns/

models.py

forms.py

views/

tasks.py # Celery email sender

imap_bounce.py # Bounce processor

recipients/

templates/

static/

🗄 **Database Models**

Campaign

- name
- subject
- content
- status
- scheduled_time

Recipient

- name
- email
- subscription_status
- groups (M2M)

CampaignRecipient

- campaign
- recipient
- status (sent/failed)
- fail_reason
- sent_at

BounceRecord

- campaign
- email
- reason
- processed_at

🔁 **Core Workflow**

1️⃣ **Admin creates a campaign**

2️⃣ **Upload recipients via CSV**

3️⃣ **Assign recipients to campaign**

4️⃣ **Celery Beat triggers scheduled campaigns**

5️⃣ **Celery Worker sends emails (SMTP)**

6️⃣ **IMAP Processor fetches bounce failures**

7️⃣ **Dashboard updates live stats**

8️⃣ **Admin downloads reports**

⚙️ **Installation Guide**

1. Clone the Repository

git clone https://github.com/<your-repo>/bulk-email-system.git

cd bulk-email-system

2. Create Virtual Environment

python -m venv venv

venv\Scripts\activate # Windows

3. Install Dependencies

pip install -r requirements.txt

🛠️ **Environment Variables**

Create a .env file or update settings.py.

SMTP

EMAIL_HOST=smtp.gmail.com

EMAIL_PORT=587

EMAIL_HOST_USER=your_gmail@gmail.com

EMAIL_HOST_PASSWORD=app_password

EMAIL_USE_TLS=True

IMAP Bounce

IMAP_HOST=imap.gmail.com

IMAP_PORT=993

IMAP_USERNAME=your_gmail@gmail.com

IMAP_PASSWORD=app_password

IMAP_USE_SSL=True

Database Setup

python manage.py makemigrations

python manage.py migrate

Create Admin User

python manage.py createsuperuser

Start Redis

Celery requires Redis.

redis-server

Start Celery Workers

Worker:

celery -A core worker -l info

Scheduler:

celery -A core beat -l info

Run Django Server

python manage.py runserver

How Sending Works

1. Campaign reaches scheduled time
2. Celery worker fetches recipients
3. Emails are sent one-by-one
4. Log entry is saved in CampaignRecipient

5. Campaign status updates automatically

📩 Bounce Processing Logic

1. Celery Beat triggers bounce checker
2. It logs into Gmail via IMAP
3. Searches *"Delivery Status Notification (Failure)"*
4. Extracts:
    - Failed recipient
    - Reason
    - [CID:x] → Campaign ID
5. Saves to BounceRecord
6. Dashboard updates failure count

📊 Reports

Bounce Report CSV

Downloaded from:

/bounces/download_csv/

Campaign Summary

- Total recipients
- Sent count
- Failed count
- Failed details
- Sent to admin via email

📦 Sample CSV Format

name,email,subscription_status

John Doe,john@example.com,subscribed

Priya,priya@gmail.com,subscribed

📈 Scalability Notes

- Batch sending via Celery

- Parallel workers for large lists
- Bulk inserts for CSV processing
- Bounce IMAP polling every 5 mins
- Can scale horizontally using RabbitMQ + multiple workers

Here for task purpose I used my test mail id and app password In production We can change /use by env variable and we can use any mailing services I used SMTP