



Banzai Technical Challenge

This Technical Challenge is about Development Python Django web Application that should have features like bellow:

1. Create a view where a user can upload an excel file with a list of contacts.
2. The excel file should have Name, Phone Number, and Email Address columns.
3. When a user uploads a file, the contacts with a phone number should be stored in a model and any contacts without a phone number should be ignored.
4. When processing the list of contacts, the same email address or phone number cannot be uploaded in a time window of 3 minutes. After 3 minutes have passed, a contact with that email or phone number can be uploaded.
5. Upon upload, the app should thank the user and notify them that the upload is underway.
6. The file should be processed in an async celery task.
7. The original excel file should be stored in S3.



Narvik Aghamalian @ 19/12/2020



Prerequisites

The application using Python/Django Web Framework as our main framework and use its dependencies and prerequisites is listing below:

Django

This framework based on Python and provide rich feature and module that help us to building, testing, securing, and deploying our application.

Celery

Celery is one of the powerful python library to help us improve the performance of our application. *Celery is a simple, flexible, and reliable distributed system to process vast amounts of messages, while providing operations with the tools required to maintain such a system.*

Pandas

Pandas is a python library that used by Data Scientist to processing the large dataset and data analytics. Pandas has rich feature that allow us to load data from several document format such us Microsoft Excel 2007, Microsoft Excel 2010, and another unlisted format.

Openpyxl

Openpyxl is a python library that allow us to modify Excel file.

celery-progress

Drop in, dependency-free progress bars for your Django/Celery applications.

Installation

Virtual Environments

Virtual environments are an indispensable part of Python programming. They are an isolated container containing all the software dependencies for a given project.

To create Virtual Environment:

- `python -m venv banzaitc`

to activate venv :

- `source env/Source/activate`

To install all Packages that used in the project :

- `pip install -r requirements.txt`

Configure application settings

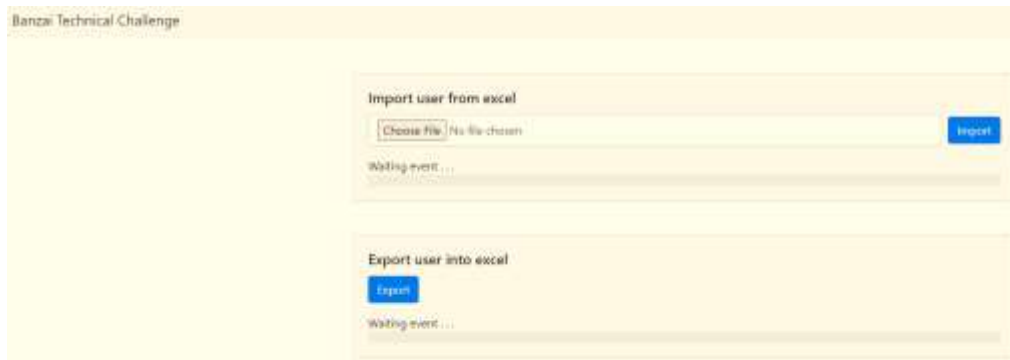
In this project we use five app and each one has its special functionality.

```
INSTALLED_APPS = [  
    'django.contrib.admin',  
    'django.contrib.auth',  
    'django.contrib.contenttypes',  
    'django.contrib.sessions',  
    'django.contrib.messages',  
    'django.contrib.staticfiles',  
  
    # Celery Progress  
    "celery_progress",  
  
    # Django Extentions  
    "django_extensions",  
  
    # Apps  
    "corsheaders",  
    "core",  
    'contact',  
]
```

Access and Admin Area

Contact

<http://127.0.0.1:8000/contacts>



Banzai Technical Challenge

Import user from excel

Choose File | No file chosen

Waiting event ...

Export user into excel

Export

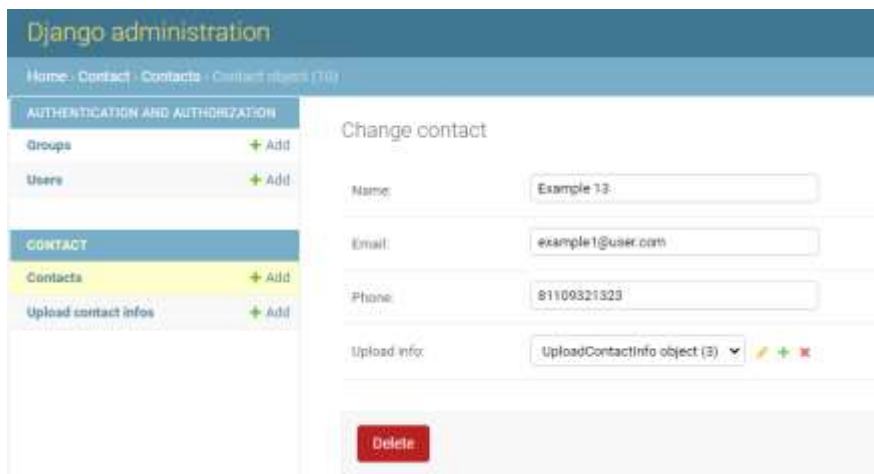
Waiting event ...

Admin Area

<http://127.0.0.1:8000/admin>

User : admin

Pass : admin



Django administration

Home | Contact | Contacts | Contact objects (10)

AUTHENTICATION AND AUTHORIZATION

- Groups + Add
- Users + Add

CONTACT

- Contacts + Add
- Upload contact infos + Add

Change contact

Name: Example 13

Email: example1@user.com

Phone: 81109321323


Upload info: UploadContactInfo object (3) [Add] [Delete]

Delete

celery integration

Celery is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation but supports scheduling as well.

Narvik Aghamalian @ 2020



If we want to execute some piece of code in python asynchronously or If we want to schedule code to execute in the future. We can use celery.

Consider below scenario:

If user uploads an excel of 5000 rows to create a database record for each row. It is a time-consuming task. The user needs to wait until the task gets completed. It is not ideal to make the user wait and the browser will send timeout error if the response is taking too long. In this case, we should execute the time-consuming task asynchronously.

A message queue is a data structure, or a container - a way to hold messages for eventual consumption. A message broker is a separate component that manages queues.

Celery require a message broker backend to handle queuing the message, store the data, and caching the data. There are many message broker backend already used and maintained by big company such us Redis, RabbitMQ, Amazon SQS, Zookeeper and others.

This project, used Redis as message broker backend because easy to install, configure, and use by big company such us Github, StackOverflow, Pinterest, Twitter and etc.

5. Redis

In this Project we use Redis Broker, you can run it from `redis/redis-server.exe` (windows OS)

User Interface for export and import

For Front End we use a simple bootstrap template that user can upload or export Excel file.

Our celery tasks

We have two main feature thats require to run on another thread.

Export Data

Import Data

Running Django and Celery server

The both application need to run the either server. Django server to run your web application, and celery server to listen your message event request from registered task.

Command to run Django server

```
python manage.py runserver
```

Command to run celery server

```
celery -A banzaitc worker --loglevel=DEBUG
```

And Other useful Descriptions..

- Importing excel file to contact table in 'tasks.import_task'.
- Rewrited exporting contact table to excel file in 'tasks.export_task'.
- Added AWS class in utils to upload files to aws s3.
- Used AWS class in 'contact.views.import_contact_view'. codes are comented.
- Added 'is_valid_dataframe' func in 'core.utils.DataframeUtil' to check file validation. used in 'contact.views'
- Added more status exceptions to notify client if something went wrong.
- Writed test cases in 'contact.tests' to test models.
- Writed doc strings on each class and func.
- Added 'env_variables.txt' that contains secret variables of AWS S3.