Documentation File

This is a follow up documentation file based on the README document found on GitHub repository at https://github.com/JuanDM93/sps_django.

API Service

This is an api demo built on Django's rest framework. It serves sample data from a hosted mongodb database and it has default auth capabilities enabled designed to scale as needed.

Service Endpoints

Once deployed, API serves the following endpoints:

Admin Panel	
/admin	It is a default login interface for CRUD actions on the api resources
Documentation	
/docs	This is a swagger based user interface
/redocs	A ReDoc based interface
Main Services	
/api	All managed endpoints are routed here
Users	
../auth	All this services require JWT credentials for session control
../../register	POST and create login credentials
../../me	GET/PUT logged User data
../../login	POST to create token
../../token/refresh	POST to refresh token
Resources	CRUD capabilities
../customers	
../accounts	
../transactions	
Health Check	
/ht	TODO

Deploy Options

There are some different approaches to deploy this service: as a local application, a dockerized container, a swarm cluster or a cloud hosted api. Just clone the repository and build your solution accordingly.

```
$ git clone https://github.com/JuanDM93/sps_django.git
$ cd sps_django/
```

NOTEs:

- As it uses a hosted DB, it is needed to grant network access for local deployments. For more details go to the ATLAS documentation page.
- For local implementation, either local environment or docker builds, an *.env* file is needed to be created at sps_demo*/sps_demo/* dir, following *.env.example* file requirements.

```
$ cd sps_demo/sps_demo/
$ cat .env.example
```

```
DEBUG=

ALLOWED_HOSTS=

SECRET_KEY=
DB_CLUSTER=

DB_USER=
DB_PASSWORD=
```

.env.example

Local Setup

Follow the following steps in order to build this solution in your local environment.

First, having Python installed, create a virtual environment at the root of the cloned repo and activate it.

```
(master|✔) $ python3 -m venv venv
$ source venv/bin/activate
```

Now, move into the app directory and install requirements.txt as follows:

```
$ cd sps_demo/
```

```
$ pip install -r requirements.txt
```

Now it's ready to run, use *manage.py* file to start the server, make migrations or other functions stated in here. This service runs by default on 8090 port throughout this documentation paper.

```
 $ python manage.py runserver 8090
```

Dockerizing

In order to build a docker image to deploy containers, follow the next steps.

Once you have the cloned repo, you can use the *DockerFile* in sps_demo/ dir to build your custom image.

```
$ docker build sps_demo/
```

Alternatively, a *docker-compose.yml* file is available to deploy *gracefully*.

```
$ docker-compose up
```

Kubernetes

For a kubernetes deployment, it is first needed to have either a custom local image built using dockerizing steps before or just use the *deployment.yaml* file found in repo which pulls a public image of the solution from dockerhub.

```
$ kubectl apply -f deployment.yaml
```

Cloud Host

For this example, we'll be using fly as a hosting service. For deployment, a docker image is also required. As with docker-compose and kubernetes, this repo has a *fly.toml* file to deploy automatically using the hosted docker image.

```
$ fly deploy
```