Backend Engineer - Product Importer - Advanced

Difficulty level: High
Skills tested: Ability to handle long running processes, use python, use ORMs and deployment

Objective

Acme Inc needs to be able to import products from a CSV file and into their SQL database. Given there are half a million products to be imported into the database, ACME Inc needs this to be done on a pretty UI that their sales crew can use.

Specification

STORY 1:

As a user, I should be able to upload a large CSV file of 500K products (see here) to the app. If there are existing duplicates, it should overwrite the data. Deduplication can be done using the SKU of the product. SKU is case insensitive. Though not in the CSV file, some products should be active and others should be inactive. The SKU is expected to be unique.

STORY 1A:

When the user is uploading, the user should be displayed a live stream of what is happening. Our recommendation would be to use SSE to implement this.

STORY 2:

After I upload the file, I should be able to view all of the products, search and filter them. This is preferably on a URL like `/products`. This view should also show a filter to see just the active products and inactive products.

STORY 3:

As a user, it should be possible to delete all existing records and start a fresh upload.

STORY 4:

As a user, it should be able to add/update product manually from UI.

STORY 5:

As a user, it should be able to configure multiple **webhooks** which should be triggered when product is created and updated manually from UI(not from csv import).
Note: Design should be scalable and should not impact application performance.

Toolkit

Acme Inc. is also opinionated about their tech stack. The tools should be:

1. Web framework: Python based frameworks - flask, tornado, bottle, django
2. Asynchronous execution: Celery/Dramatiq with RabbitMQ/Redis
3. ORM: SQLAlchemy (if not django).
4. Database: We recommend postgres (and works with the deployment choice below).

5. Deployment: Heroku is the preferred platform to deploy this. Basic version of heroku is free and you should be able to host it there. Python Anywhere, AWS, Google Cloud, Digital Ocean or any other cloud provider are all acceptable (but of-course takes more work).

Next Steps

- Build the app
- Push the code to Github/Bitbucket/Gitlab
- Deploy the app
- Send us the link to the app

Points and Rating scheme

APPROACH AND CODE QUALITY

Code quality is a very important part of the assignment. Better documented, standards compliant code that is readable wins over brilliant hacks. Remember CPU and Memory are cheap and expendable, humans are not.

COMMIT HISTORY

Clean commits and a good commit history offers a sneak peak into planning and execution.

DEPLOYMENT

Gone are the days when a different team deployed to production. Engineers maintain infrastructure as code and being able to deploy your own little app is a great way to see how something you build works in the real world. The easiest way to achieve this would be to deploy to a PaaS like heroku.

TIMEOUT FOR LONG OPERATIONS

Heroku has a 30 second timeout. The product upload is expected to timeout and getting around this limitation requires the use of asynchronous workers and there are multiple ways to achieve this. Your solution must address that elegantly. Ask us if you need ideas.