Group 4 report for assignment 1 - File processing web service

Railway link:

https://web-file-processor.up.railway.app/

API system functionality and structure

The main functionality of this API is how it receives either a “. tar.gz” file containing the necessary files, or all separate files, then proceeds with processing them. Finally, the processing generates a downloadable file called “certificates.tar.gz” containing individual PDFs for each one of the people found in the .CSV file. The necessary files can be found within “Dummy” folder under /src, and can be used to test the system.

In-depth about the processing:

1. Function “read_csv_file()” reads the uploaded csv file.
2. Function “modify_and_write_markdown()” takes the uploaded template markdown file, and modifies it with the data from the csv file. First name and last name are replaced with the actual names.
3. The new markdown files (one for each person) are added to the “MD” folder.
4. Then, these MD files are used in the “convert_to_pdf()” function, and a PDF for each person is generated and added into a PDF folder.
5. Finally, the function “create_tar_file()” is used to take the PDF folder containing all PDFs and compress it into a “.tar.gz” file. This file is added to a folder called “processed”.

API routes:

- The index route “/” simply renders the index.html containing the form.
- The upload route “/upload” handles the interaction when files are uploaded. When they are, the file processing is initialized and will output the final “. tar.gz” in the processed folder.
- If the upload is successful, the “/upload/success “route renders a new page “upload_success.html”. Here, the file(s) from the processed folder are available for download, and there is also a link “upload more” which redirects the user back to the index page.

- Route “/download” handles the actual download. The “. tar.gz” file is sent from the processed folder as a downloadable attachment when the user clicks the “download file” link.

Decisions during development

During the development process, we made decisions to ensure the functionality of our system.

- We chose Flask to help build out our API. It is considered a lightweight and flexible framework and we found it to be a good choice for our project.
- We selected the FPDF library for generating PDF files from markdown due to its simple and useful features.
- We decided to use the flake8 library, a python linting tool to test for errors and styling issues.

How to use the system

Here are the few steps how to use the system:

1. Click on the upload button to select the file(s) you want to upload.
2. Click on the upload button to initiate the process.
3. After processing is complete, you will have options to download the processed files.
4. Click on the download link to retrieve the processed files to your local device.

How to setup a system like this

1. Setting up dependencies with flask and FPDP:
   To set up dependencies with Flask and FPDP, install all necessary tools and libraries, such as flask and FPDP.
   pip install flask.

2. Code testing for reliability:
   Ensure that the code is reliable, and functions as expected, conduct testing of the application, including unit testing of individual functions.

3. File processing functions:

Develop functions to handle file processing. We need functions for tasks such as reading CSV files, modifying the MD template, generating PDF files, and creating compressed archives. We used the python libraries os, csv, and tarfile. And these will form our applications' functionality.

4. Application testing in action:
   Once we implemented our code, we test it and upload files, process them using our functions, and download the processed files to ensure everything works as expected. These steps are essential for identifying any bugs or issues that need to be fixed. Testing was one of the most challenging parts of the assignment. We made multiple attempts to make automatic unit testing work, but not all the issues were solved. Due to our unfamiliarity with testing in general, we could not quite pinpoint the cause. This is regrettable, as testing is important to ensure everything works as expected in multiple different environments.
5. Deployment to Netlify vs Railway
   Finally, once our application was mostly done, our intention was to deploy it with the cloud provider Netlify. We tried to use Netlify at first but discovered that Netlify is better for static websites, and we could have to create so called flask functions in that case. Therefore, we decided to use Railway since it was easier and more compatible with flask. We attempted to deploy it with the Unicorn library since that is the better choice, but since we ran into errors, we decided to go without it.

How to replace file processing with another use case

To replace file processing with another use case, you need to modify the file extensions and adjust the processing functions accordingly. This involves updating the supported file types and path to align with the requirement of the new use case.

Group 4 members:

Sabrina Altahrawi
Maja Celin Brunsvik
Elina Kati Nystø Keskitalo