An Open-Source Personal Finance Manager

Background:

Personal Finance Manager (PFM) applications have been around since the advent of the PC. Recently, these applications have migrated online, in the form of popular sites like Mint. Mint and similar apps aggregate users’ income and transaction data, and provide a host of visualization and analysis options. Moreover, many of these sites have additional features which allow users to set goals and budgets, manage alerts on accounts, and find potentially hidden spending trends.

These features, and the entirety of the app, depend on the reliable collection of data. Some apps require the user to enter bank login information, and then rely on advanced scraping techniques to collect information. Others, especially self hosted versions, force the user to manually download their transaction data and upload it to the application to be processed. Some require manual entry of every transaction, and rely on the user’s memory for accuracy.

While scraping is the most convenient for the user, it is computationally expensive, and often violates the ToS of the banks. On the other side, the load from repeated scraping tasks poses significant problems for the bank’s servers. Lastly, because updating and fixing scrapers is in itself a huge work-effort, many PFM’s delegate this duty to companies like Yodlee, which charge high prices for the service.

From the privacy standpoint, many users are not comfortable with providing these services with their account information. Security risks are especially high because ids and passwords must be retrieved in plaintext to login to the sites.

PFM’s that rely on users uploading documents have even more obvious problems. Downloading and reuploading the files is a significant burden on users, especially when often the file only differs from the previous version by one or two transactions.

Finally, some users may not trust external services with any financial information. This makes an open-source self-hosted version especially valuable.

Proposal:

As far as I've researched, I could not find a solution that satisfies all 3 of the following properties.
  1. Does not require bank account information.
  2. Automatically updates transactions.
  3. Open source and can be self-hosted.
1. Every major bank or credit card firm provides an “Alerts” feature to their user. This feature allows users to set conditions upon which they will receive email messages about account activity. With transactions, for example, I can ask to receive an email on every transaction. This project will use this feature, by either receiving the emails directly, or through a forwarding filter, to get collect data. Thus, the app is completely unaware of any important bank information. Moreover, the push vs pull model places a majority of the resource intensive work on the banks themselves without artificially increasing traffic by scraping. Lastly, maintaining parsers for plain text emails is much simpler than maintaining complicated Javascript enabled scrapers.

2. Because new transactions will all send alerts, users do not have to upload files or enter data by hand. However, the app will also support importing bank files if the user wants historical data available.

3. The entire application, back-end and front-end, including the mail receiving and processing portion, will be completely open source with instructions for self-hosting. The app will be modular enough that solutions that don’t require a mail-server can be swapped in for data updating.

Features:

- Transaction listing, searching, and modifying - Users should be able to search through transactions using different filters, group them into categories, and use conditions/regex to modify recipient names (ie. /*Yale*/i -> Yale University) intelligently in all previous and future data.
- Users should be able to chart spending using different visualization tools.
- Support for other bank account activity (transfers, paychecks etc.)
- Users should be able to set budget goals and track progress.
- Supports importing of Open Financial Exchange format files.
- Supports multiple users.

Constraints:

I will also be building the application with Clojure/ClojureScript, a language to which I’ve had very minimal to no prior exposure. Unfortunately, I have not had the opportunity to work with functional programming except in two classes, and I think it would be an amazing learning experience to use Clojure for a larger project.

I will also make as much of the project as possible testable. While I’ve worked with testing before, it will be interesting combining unit tests with the integration and functional tests needed for this kind of application.
Deliverables:

Basically everything in the Features+Constraints section:
- A modular, testable, working application with at least 2 different bank sources.
- Forward and direct delivery options for mail updates.
- Aesthetically pleasing UI - I am not a designer but I would like to challenge myself here.
- A better name.

Motivation:

I think this project would be a great opportunity to learn about many different areas of CS and software development which I haven’t spent too much time with: functional programming, intelligent document parsing, category indexing, data storage with serious security implications etc., mail servers, front-end development etc.

Additionally, I think this is an applicable and reasonably-scoped project which people could use--I would definitely use it.