

Team: JaJeJu

Roster: Jackson Zou -- Jeff Lin -- Jun Tao Lei

Project: Spendie

P2

## **Summary**

The premise of this project is to design a site that tracks one's spendings over a period of time. The site will be able to display statistics for total transactions over a month such as spending breakdowns, total spendings, deviation from spending targets, etc. There will be tags that link to the different custom categories of transactions over a month.

A calendar will be included to display transactions over a month. The site will be able to keep track of loans or debt that may be viewable to other people or keep track of certain notes or todos. Users can also search for certain customized tags.

## **Front-end Framework**

Bootstrap will be used as our front-end framework as it is the more familiar framework.

## **Components**

- Spending Statistics
  - This shows the sum of all the transactions made in a calendar month.
  - This shows the spending target and the deviation from the spending target in a calendar month.
  - This shows the spending breakdown of all the transactions made in a calendar month based on the category of each transaction.
  - This can be changed to show statistics of other months
- Tags
  - Tags refer to a unique category of transactions.

- Tag is linked to all the transactions under the specific category.
- Transaction
  - Each transaction is a collection of data including the transaction name, the transaction amount, and an optional note about the transaction.
- Calendar
  - The calendar can track all transactions made in a calendar month
    - Each day contains the number of transactions made and the total amount of the transactions.
    - The color of each day corresponds to the percentage of that day's spending as a percentage of the monthly spending.
  - Would be easier to build with external javascript framework
- Search
  - User can search for a transaction based on the note associated with each transaction.
- Todos
  - This keeps a list of todos.
- Requests
  - Give/get requests for money from other users
  - Requires specific user\_id to request
  - Will show notifications for any of these requests

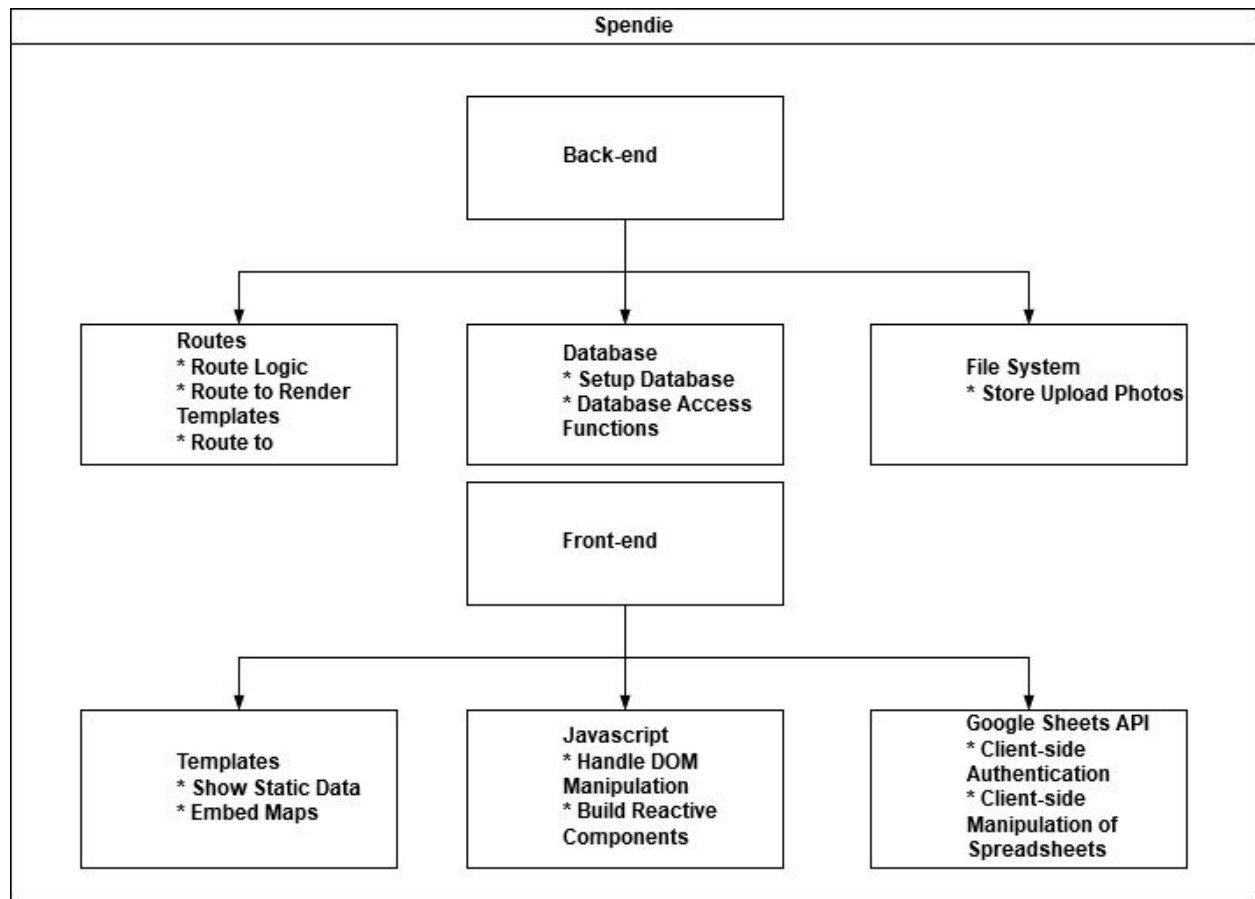
## **API Integration**

- Integration with the Google Sheets API and the Google Maps Embed API
  - The Google Sheets API allows the user to export his/her data to his/her Google account.
    - Requires Google's external javascript file to authenticate, create, and manipulate spreadsheets client-side.
      - <https://apis.google.com/js/api.js>

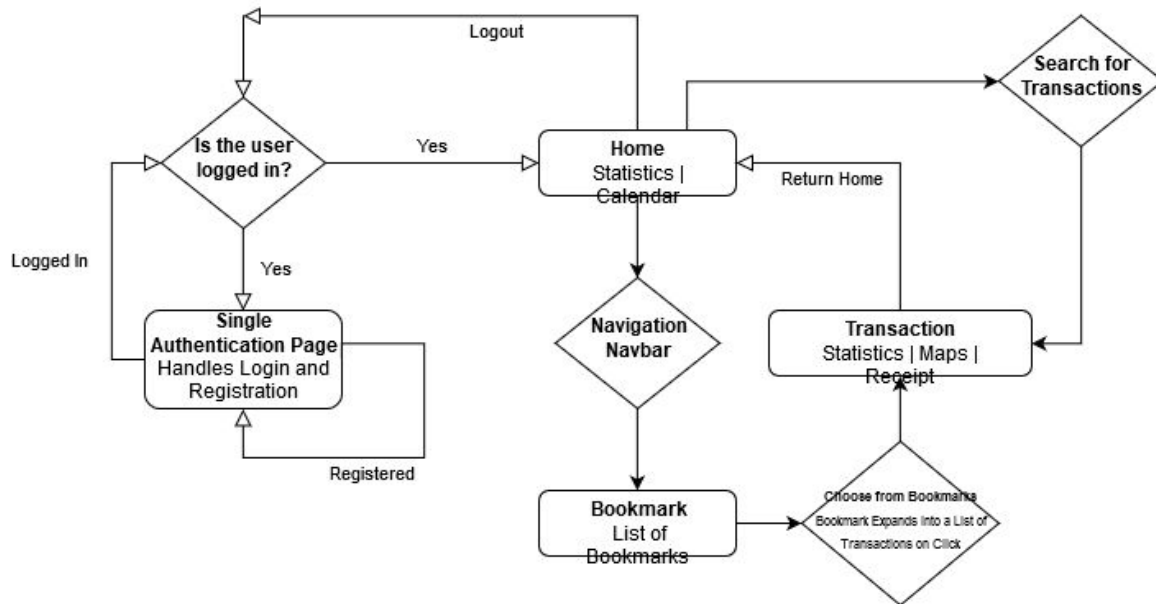
- The Google Maps Embed API allows the user to view the location where the user made a transaction.

## Component Map

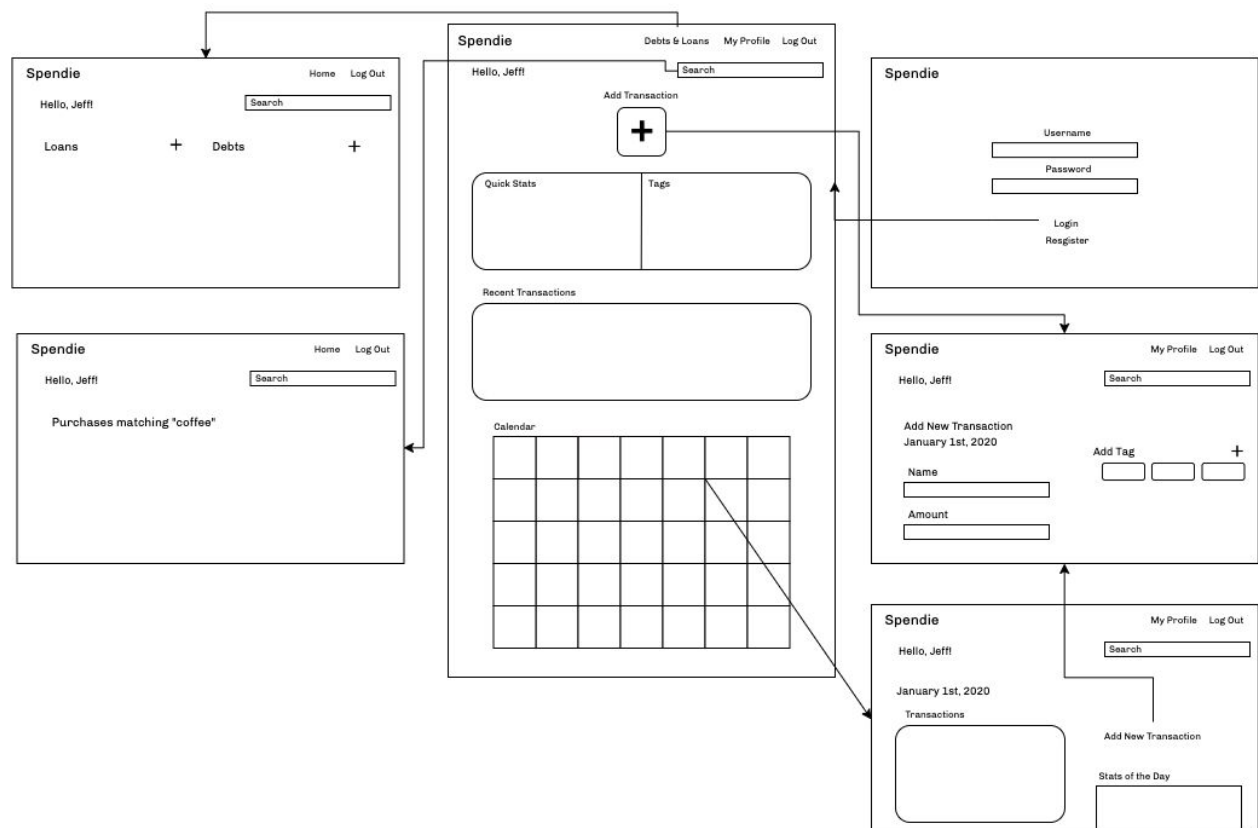
- Frontend:
  - Javascript will allow for user interaction with the site
  - Bootstrap will be used to format site and give aesthetics
- Backend:
  - Flask will serve as an intermediary between front end-user interaction and database
  - SQLite databases will store user inputs and return information as necessary



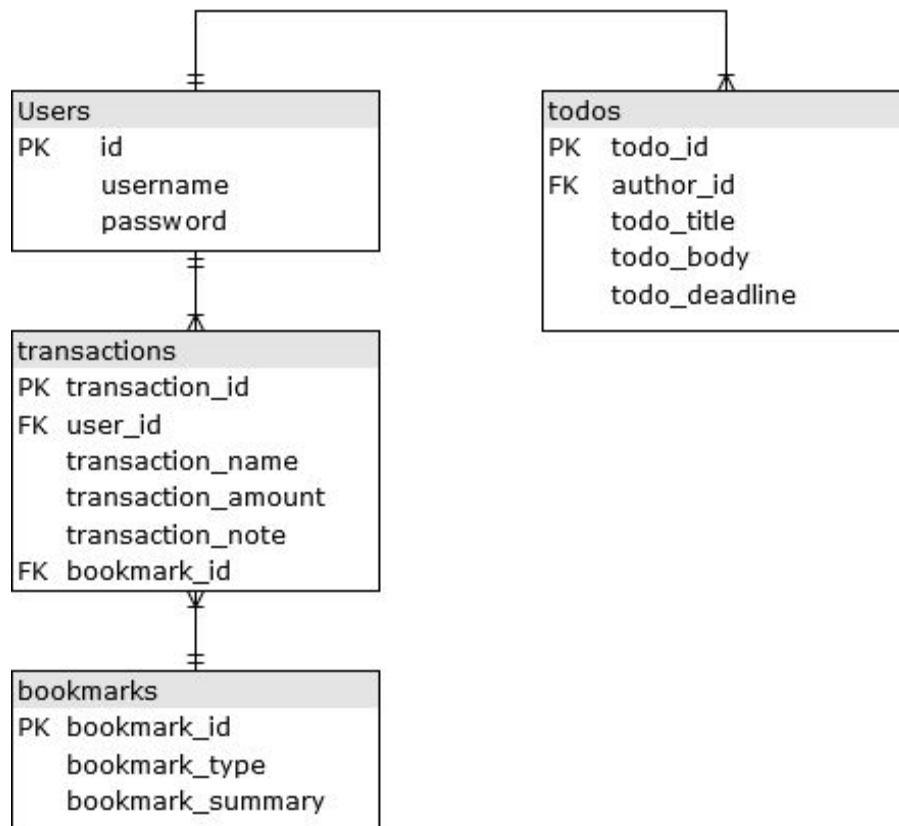
## Site Map



## Design



## Database Diagram



## Design

This application will have some distinction between the client-side and server-side functions. The application will serve server-side rendered templates and static javascript files. Besides serving some static javascript files, the application will be monolithic.

For the most part, the application front-end will serve server-side rendered templates for trivial operations like viewing transaction statistics. However, static javascript files will be served for interactive components like the calendar. Although a template would probably still be used for such components, the template HTML will link to the javascript files.

The application back-end will handle database access operations, file system storage operations, and authentication. Otherwise, it would likely have a similar implementation to previous projects.

While the Google Maps Embed API can be used by embedding the template with an iframe, the Google Sheets API would probably be implemented client-side. This is because it would make more sense to have Google authentication be performed client-side to avoid building Google authentication into the user flow. Therefore, it would be likely that the client would fetch data from the application back-end using Javascript, and populating the Google spreadsheet with that data.

## **Roles**

- Jackson
  - Back-end
    - Flask Routing Logic
      - Check if a user is logged in, etc.
    - Flask Routes for Transactions and Bookmarks
      - Query Database
- Jeff
  - Front-end
    - Templates and Styles
      - Embed iframe for Maps
- Jun Tao
  - Back-end
    - Setup Database
    - Setup Authentication
    - Setup Instance File System Upload
  - Front-end
    - Javascript
    - Client-side Google Sheets API