Sidework

Katy Ross
@katyross

Description

A budgeting application with the goal of promoting healthy financial habits for service industry workers. End product will allow users to track incoming tips and hourly wages, track deductions such as tip out and tax withholding, budget with one's estimated bi-weekly income, and demonstrate one's ability to pay extra towards debts to lower the lifetime of one's loans.

Features

- Income
  - Enter daily hours, cash tips, and cc tips
  - Calculate deductions based on restaurant tip out rate and shift's food/bev sales
  - See estimated income based on payday, between certain dates, restaurant, or all
- Budget
  - Add edit and remove expenses
  - Indicate whether an expense is a recurring monthly or variable amount and/or a debt
- Debt Payoff
  - Display extra funds available to pay off debts
  - Display, delete, and edit debts
  - Choose between avalanche and snowball methods of debt payoffs
  - Calculate debt payoff date based on a number of variables

Planning - User Stories

“As a server, I want to add multiple restaurants so I may log several streams of income.”

Hospitality professionals often times have multiple positions at one or several restaurants, which can make tracking income messy when you factor in different rates of pay and different tip-out rates. With this in mind, the ability to add and edit workplaces to account for these variations was implemented.

Planning - Database

- Restaurants
  - id , bar tip-out %, food tip-out%, hourly rate, and name.
- Shifts
  - id, bar sales, cash tips, cc tips, food sales, in time, out time, payday, and restaurant id.

When the user creates a new shift, they are prompted to select a restaurant they've saved, that restaurant's id is saved in the shift object and called when the user runs an income query.

Technology Stack

- Java
- Angular
- TypeScript
- Spring Boot
- MySQL
- Bootstrap

Demo

What I Learned

- Was familiar with Angular, but grew my skill-set exponentially while implementing specific features I needed. Much more comfortable with TypeScript as a result.
- Rest API - I learned how to get the front-end and back-end to talk!
- Using input binding to pass data between parent and child components
- CSS IS HARD

What's Next

- Budget, Debt PayOff, and User login components!
- Before I can start on those…
  - Add validation to front-end forms
  - Fix close click issues on View Income modals
  - Global CSS so additional styling is unnecessary unless for specific needs
  - Format tables, and modals. Add minimal animations for certain buttons
  - Clean up front end