From Excel to MERN

Creating a Data Matching Web Application

Shira Abramovich
Civic Digital Fellow, Bureau of Labor Statistics
Office of Employment & Unemployment Statistics (OEUS)

CIF Summer Presentations

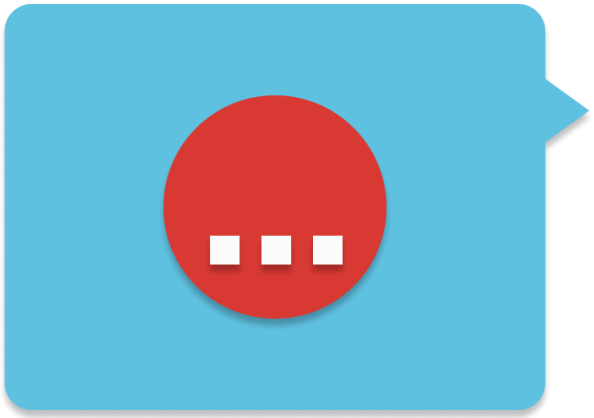
August 4th, 2021

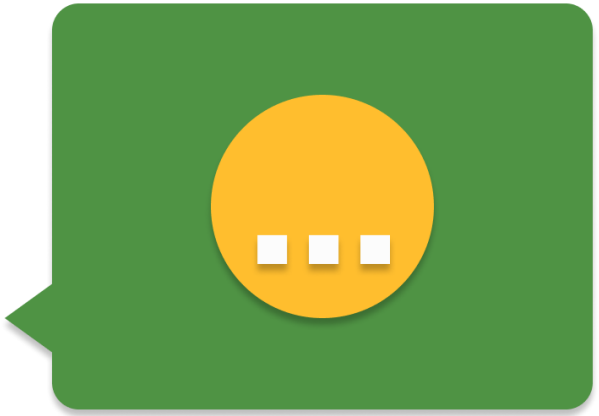
Background

- Question: how much/how many U.S. employment, wages, & occupations are foreign-owned?
  - Foreign-owned: “at least one foreign owner with at least 10% ownership” (Source: BLS FDI home)
- Answer: ???
  - **No single existing data set** to analyze question
    - BLS collects **domestic** data via QCEW
    - BEA collects **foreign investment data** via Survey of Foreign Direct Investment (FDI)
- Idea: **record linkage!**

Background: Record Linkage

- Idea 1: join QCEW & FDI data sets on basis of **Employer Identification Number (EIN)**
  - …but EIN-based linkage → **87.7% initial error rate!**
    (Source: "Using Entity Resolution to Reduce the Cost of Producing Inward FDI—QCEW Estimates")
  - Problem: EINs are problematic!
    - may change over time due to changes in company
    - company may have multiple EINs
      - different people may fill out QCEW & FDI → may report different EINs to different surveys
- Idea 2: use a **record linkage algorithm** to take into account other attributes (e.g., name, contact information, etc)
  - Problem: algorithm still needs human review to verify accuracy & correct disparities





Background: Human Review

- Prior solution (used with 2012 data review): Excel spreadsheets
  - Pros: Easy to view data, easy to distribute tasks to individual analysts (just copy over rows), **low barrier to entry**
  - Cons: ~~Excel spreadsheets~~
    - Difficult to create & keep track of tasks (requires lots of administrative overhead)
    - Too easy to accidentally delete data
    - Difficult to visualize data
    - Takes a long time - nearly **800 hours** for full match (Source: "Using Entity Resolution to Reduce the Cost of Producing Inward FDI—QCEW Estimates")
    - Costs lots of money
- New solution: **web application**

Initial Spec

"A **web-based User Interface (UI)** to our record linkage system….

In order to review a particular set of predicted linked QCEW establishments for a given FDI firm, the UI must report several metrics in **both graphical and in tabular form**….The UI should allow analysts to drill down and remove establishments that were linked in error. In addition to removing establishments that were linked in error, the analysts need to add establishments that were not linked. After all establishments that were initially linked in error have been removed, and all establishments that were not initially linked but should have been have been added, the **UI should allow the analyst to confirm that the firm is fully linked** and these results pushed back to the server….

The UI is also intended to **facilitate the administration of analyst review**. Administrators will use the UI to assign FDI firms for analysts to review. To facilitate the assignment of firms to analysts, the UI should provide similar metrics as those given to analysts but for all firms. In this way, poorly linked firms can be assigned first."

Source: CIF Project Proposal

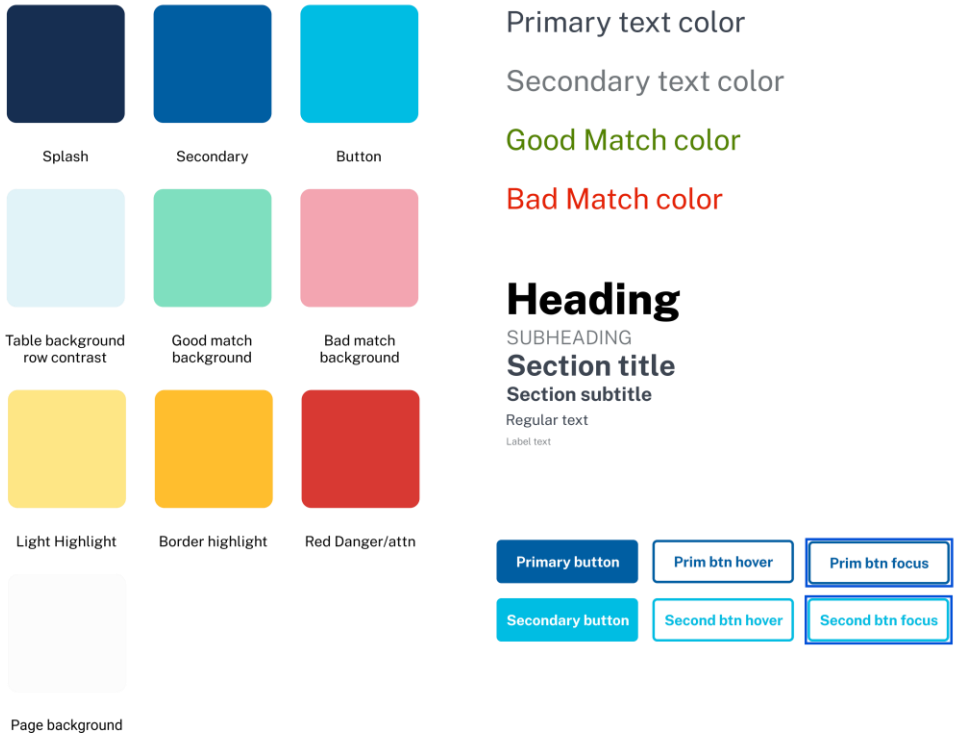
User Research

- User research: informal interviews
  - with prior administrators
  - with prior analysts of 2012 data
- Extrapolated task information & desired features from user interviews
  - e.g., administrators would benefit from infrastructure to track how much time analysts spend on a record; analysts often use name & address data to assess goodness of match

Design Research

- U.S. Web Design System
  - used as inspiration and guide, but not directly imported
    - difficult to integrate with React
- External design packages
  - help reduce overall design work needed
- Caveat: **I am not a designer!**

We want to use fonts supported by the USWDS, so we will likely go with Public Sans.



Excerpt from Figma design sheet

Technical Research

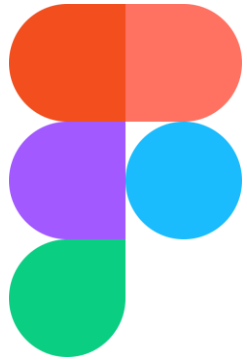
- Tech stack
  - Database: matched data was already in NoSQL→ gravitated to MongoDB fairly early
  - Front-end
    - App: looked into React & Angular; chose React based on experience level (longtime React developer; little Angular experience)
    - Visualization: looked into Tableau, D3, a few other libraries; chose D3 fairly early on as it's the gold standard
  - Server/middleware
    - Chose Node/Express due to experience & ease of use

Technical Research

- Infrastructure
  - Version control: **GitLab**
    - **essential** for management of multiple app components & features
    - caveat: tricky to work out multiple collaborators on single remote server
  - Development environment: **VSCode**
    - integrated terminal & port forwarding allow remote development
  - Design: **Figma**
    - allows for collaborative viewing & design work without Adobe subscription

GitLab logo

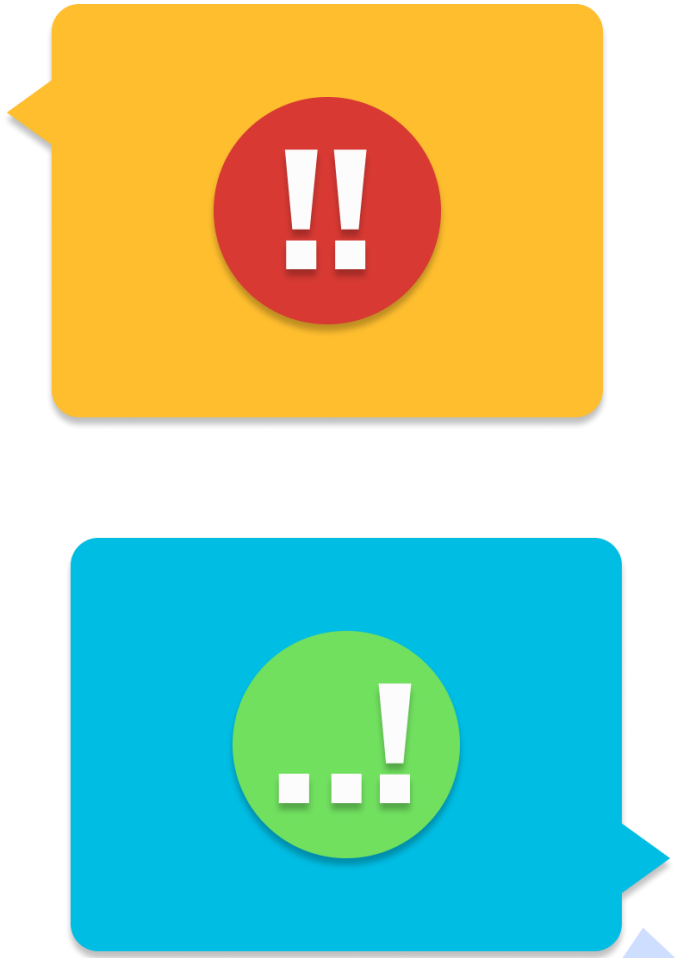
VSCode logo



Figma logo

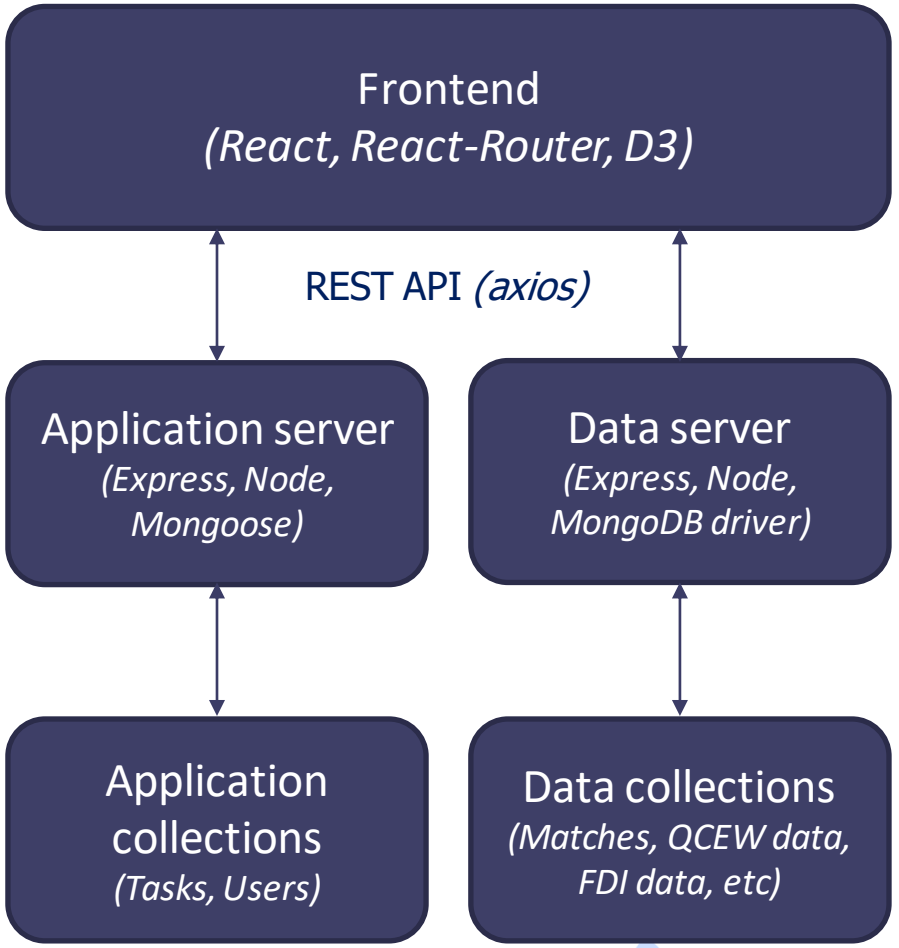
Revised project goals/MVP

- **Full-stack** prototype with extensive documentation
  - Ability to log in/log out as admins and analysts, create data review tasks, review data, “drill down” into EINs
- **Initial visualization components** in React & D3
  - Bar chart, heat map, choropleth diagrams



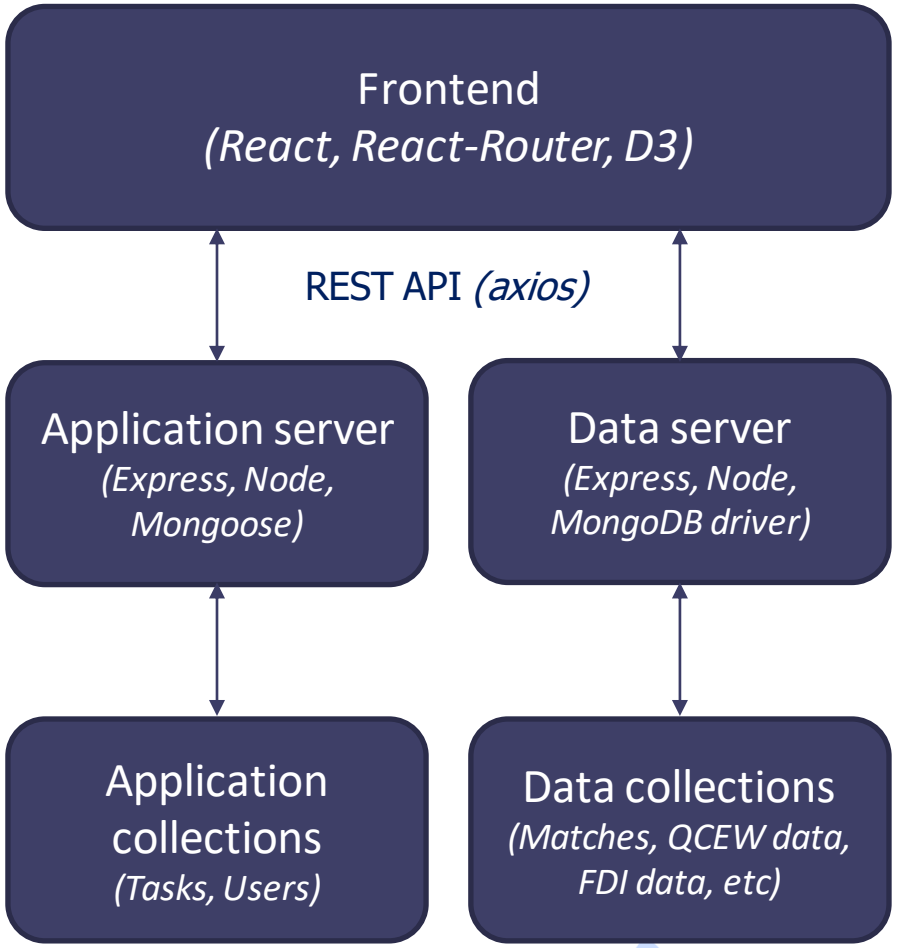
Full Tech Stack

- MERN: JavaScript across the stack
  - **M**ongoDB: Database
    - Matches existing NoSQL structure of data
    - With **Mongoose** package, constrained yet flexible structure
  - **E**xpress: Middleware
    - With **axios** package, allows management of client requests using **REST API**
  - **R**eact: Frontend
    - Powerful open-source library developed by Facebook
    - Many, many package add-ons simplify development
  - **N**ode.js: Server
    - Lightweight, JS-based server manages requests to database



Tech Stack

- MERN: perks
  - Unified language across tech stack
  - Widespread adoption → lots of open-source packages available
- Other paradigms used
  - **Microservices**
    - **Two** servers manage requests to databases
      - **App server:** manages all app-related data
      - **Data server:** manages requests to static databases
  - **REST APIs**
    - Frontend & backend(s) are **separate** for **all** purposes
    - Communication via API requests from frontend to backend(s)
      - **Frontend application initiates & resolves all actions**



Process

1. Research & design
2. Development & initial testing
   - Front-end infrastructure → back-end feature development
     - initial testing of back-end done via front-end interaction rather than scripts
     - in practice, essentially concurrent, and sometimes flipped!
   - Branch off features or feature groups, merge as features finished
     - e.g., auth branch for authentication work
3. Style & cosmetic edits

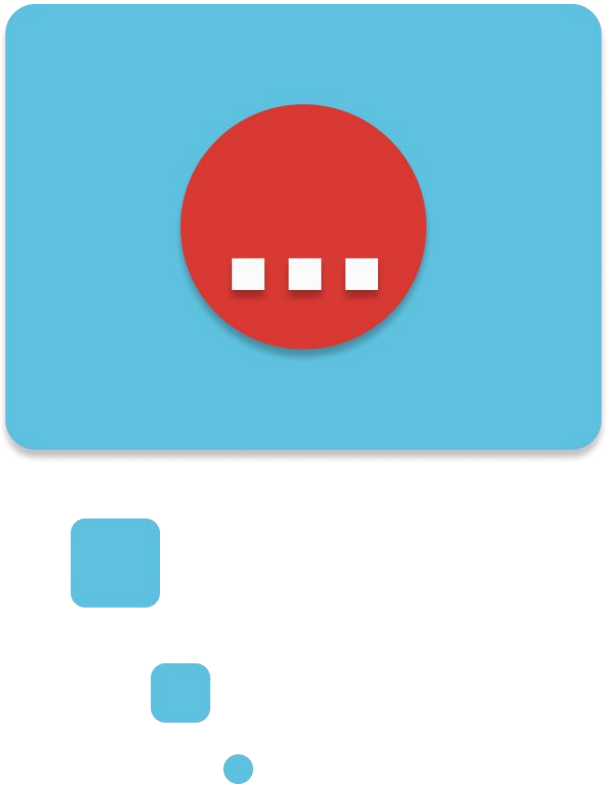
Demo

Next Steps

- Clean up existing features and patch known bugs
  - e.g., minor bug on log-out (those with keen eyes may have spotted it!); heat-map & choropleth components need wrapping up (notice you did not see them)
- Finish documenting existing work and supported future features
  - e.g., task analytics stored but not displayed in demo version

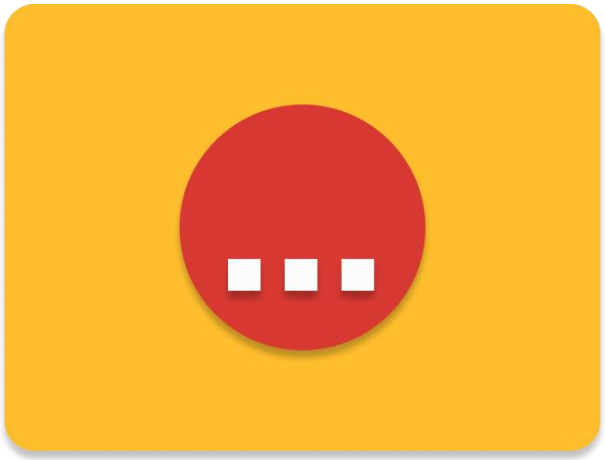
Reflection

- What would I have done differently?
  - consulted with more designers/design resources
    - this was an early bottleneck in the project, especially with visualizations
  - started thorough documentation earlier
  - scheduled more demos throughout the process



Reflection

- This was a big project – we made good progress!
- Infrastructure now exists to build out V1 of app into more robust product with broad features
- Project can serve as its own data point in creation of data-matching apps across BLS in the future!



Contact Information

Shira Abramovich
Civic Digital Fellow
OEUS
abramovich.shira@bls.gov
Supervisor: Lowell Mason
Data Scientist
mason.lowell@bls.gov