Insait Home Assignment

Akiva Buckman

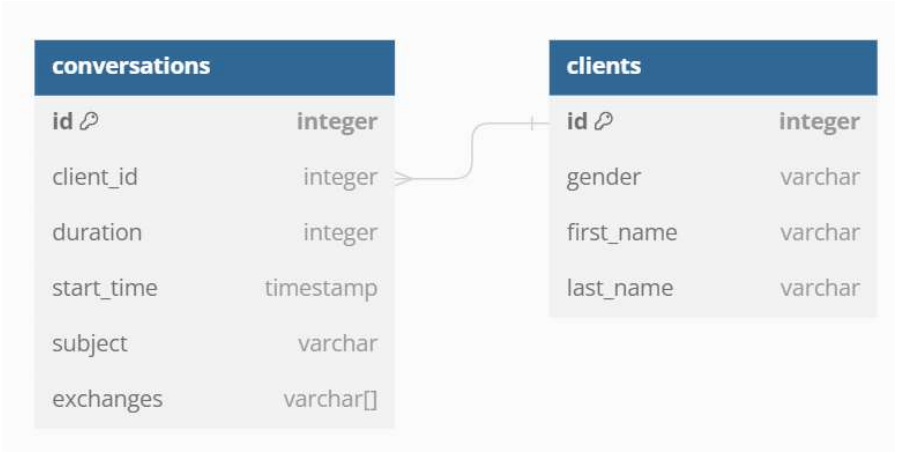
The InsaitTest project is a program that generates a database with banking-related data, populates the database, analyzes the data, and displays interactive visualizations of the generated data.

Technologies used:

- Database: PostgreSQL (relational database)
- Backend: Node.js, Express
- Frontend: React
- Other tools: Faker, MaterialUI, Knex, Nivo Charts

Database Structure:

- The project database consists of tables: clients and conversations
- The client ID in the conversations table is a foreign key to the clients table
- Database schema:



Database Population:

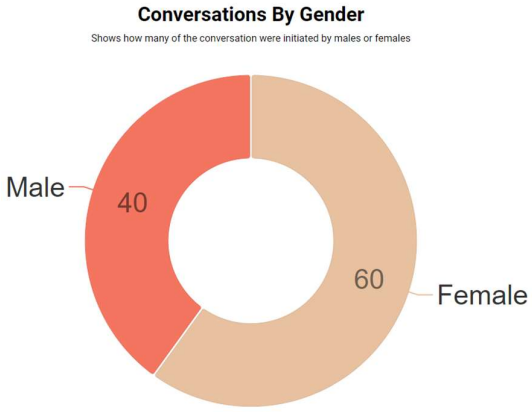
- The database is populated via a post request to the backend API (described below)
- The server contains an algorithm to classify each conversation by subject
- The client names are generated by the Faker API
- Conversations generated by ChatGPT
- Conversation start times are deliberately designed to yield a bell curve with 11:30 AM as the average

Backend:

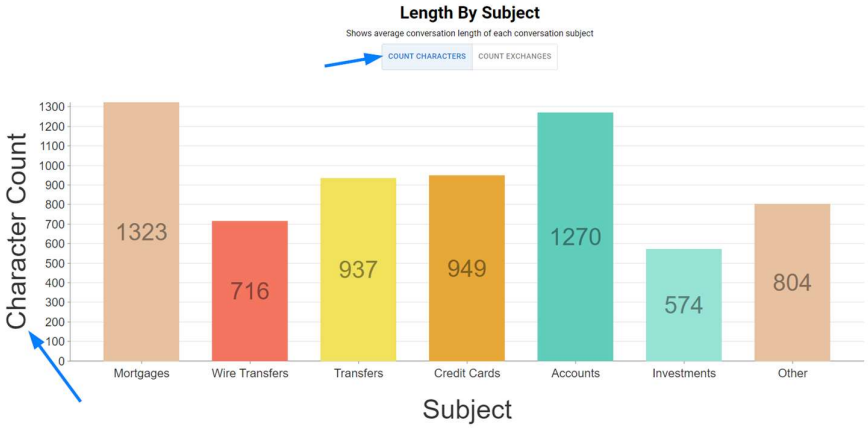
- Express server that contains 3 routes:
  - Clients – populates and retrieves client data with database
  - Conversations – populates conversations into database
  - Analytics – manipulates data from both tables to serve the client for data analytics
- Each route contains a controller and model to interact with the database

Frontend:

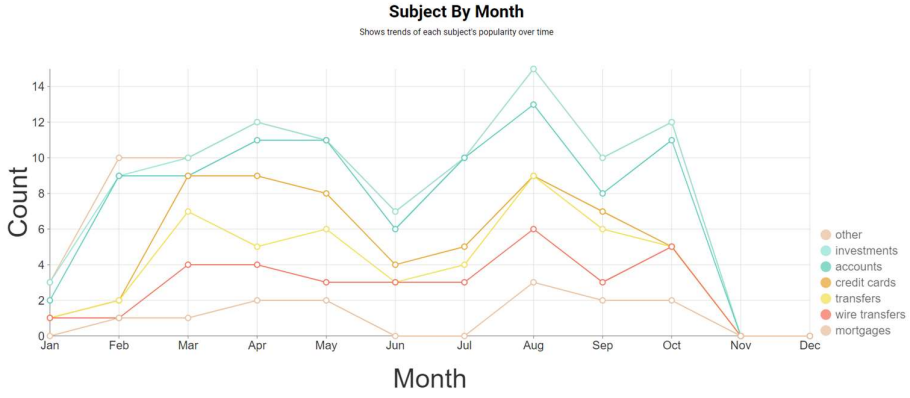
- React app that visualizes that data fetched from the server
- Consists of a Navbar and 5 analytic charts:
  1. Conversations by gender – how many conversations each gender of clients initiated
     a. In this example, the split was 40-60 male-female:



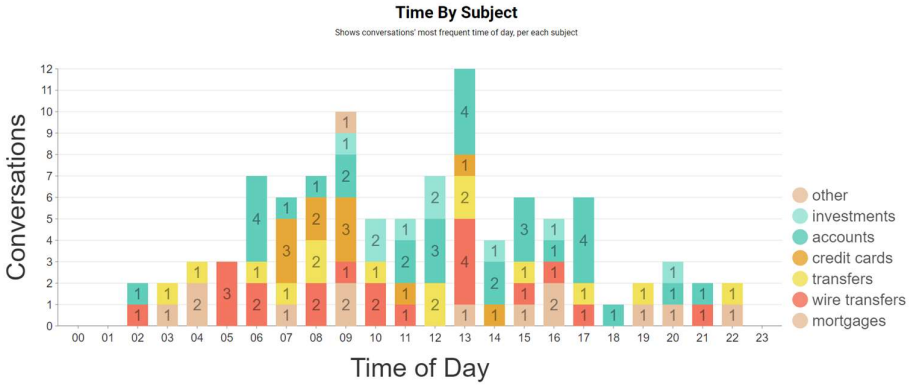
  2. Length by Subject – conversation length (toggle between characters or exchanges) of each conversation subject
     a. The number in each bar represents how many characters / exchanges were used (controlled by the toggle buttons), on average, in the conversations of each subject:



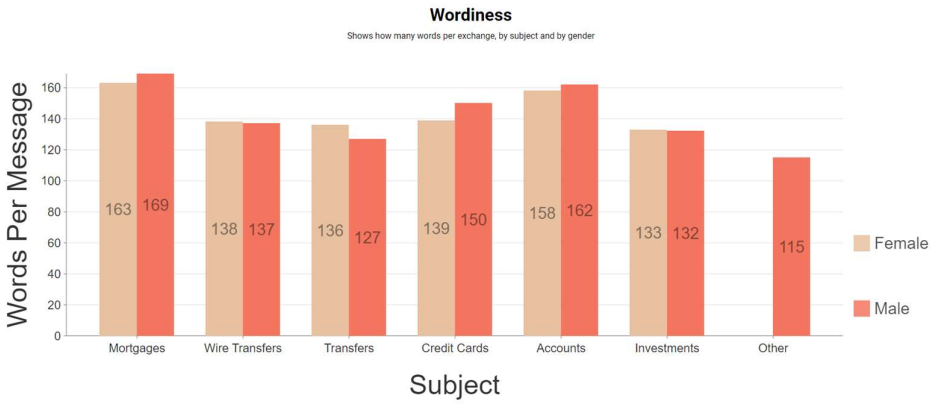
3. Subject by Month – trends of popularity of each conversation subject over time
    a. Passage of time represented by the X axis, conversation count by the Y axis:



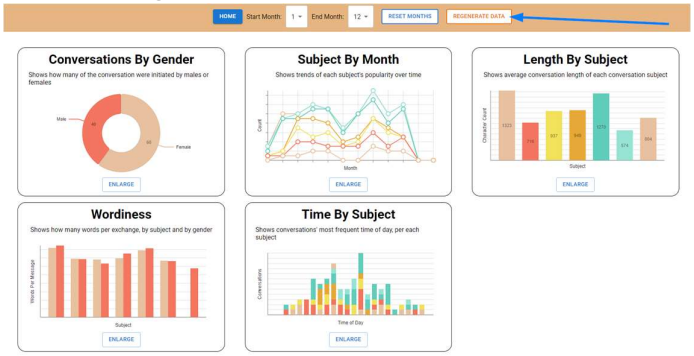
4. Time by Subject – frequency of conversation start time throughout the day, showing subject
    a. In this example, the busiest hour of the day is 13:00. This was by design in the backend that the middle of the day will be busier, for the sake of being realistic.



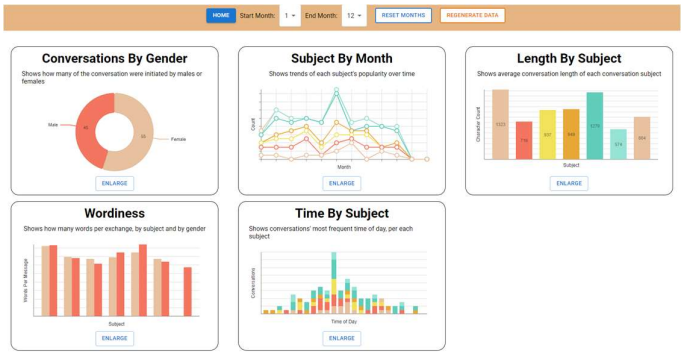
5. Wordiness – how many words in each message, in each subject, per gender
    a. In this example, mortgages conversations contained the highest average amount of words per message, for both males and females



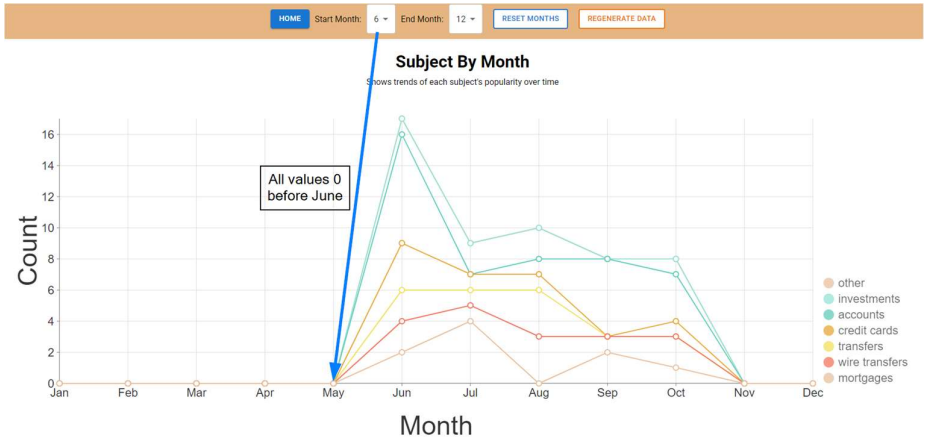
- Dashboard screen that shows all 5 charts (seen below)
- Smooth navigation between the dashboard and an enlarged version of each chart
- The dataset can be completely regenerated by a button in the navbar (which sends a post request to the backend as mentioned above). All charts dynamically, animatedly, and immediately change to represent the newly populated data
    - Demonstration:
        - Before clicking “REGENERATE DATA”:



        - After (note differences in charts):



- All charts can be filtered to a specific month range, using dropdowns in the navbar. All charts immediately adjust upon month selections.



Challenges throughout development:

- Data manipulation: due to the complex nature of the data being visualized, the SQL queries and JS code that further manipulated the data into the exact format the charts required were a good challenge
- API self-interaction in some instances, the API calls other requests within itself