RecipEat

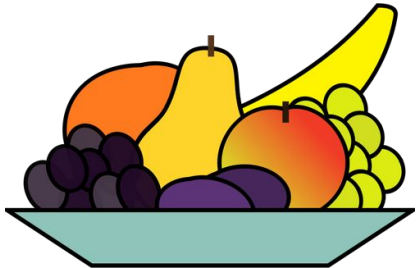
Corina Geier, Jeffrey Lai, Edward Lou, Sai Muktevi & Andrew Zhou

Outline

- Project Overview
- Use Cases
- API: Spoonacular
- Data Sources
- Demo
- Design
- Project Structure
- Lessons Learned
- Future Work

Project Overview

- Background:
  - Making healthy meals and finding new recipes can be challenging
  - Our goal was to make it easy to find healthy recipes that incorporate your nutrient targets based on the ingredients you have at home
- Our product:
  - Saves user information through login system
  - Allows a user to keep track of the ingredients they have
  - Returns recommended recipes based on ingredient and nutritional inputs
  - Provides a visual comparison of selected recipes



Use Case

- Target Audience:
  - Anyone who cooks at home and has a desire to eat healthy meals
- Use case:
  - Emily logs onto RecipEat where her bag of ingredients is saved
  - She updates or deletes her ingredients as needed
  - Next Emily uses the recipe recommender to find some recipes that meet her nutritional goals
  - RecipEat returns some recipes and Emily uses the visual comparator to see the differences in ingredients and nutrients
  - Emily is able to easily choose a healthy recipe to make

API: Spoonacular

- Spoonacular:
  - Food API that provides access to over 365,000 recipes
  - Supports ingredient and nutrient based queries
  - Filters based on food intolerances and dietary restrictions
  - Visualizes recipe information such as ingredients and nutrients
- Limitations:
  - Free version only supports 150 searches per day
  - Only 100 recipes max are returned per search

Data Sources

- Firebase for user registration and authentication.
- PostgreSQL for maintaining Bag of Ingredients data for users
- Spoonacular API data for displaying recipe results

Data Limitations

- Firebase - NoSQL. Pyrebase was difficult to use. Can’t insert, update, and query easily.
- PostgreSQL - Decoupled system with firebase.
- Spoonacular API data - Complex API search queries. We had some trouble getting requests back. Spoonacular has bad latency (Avg. 1000ms).

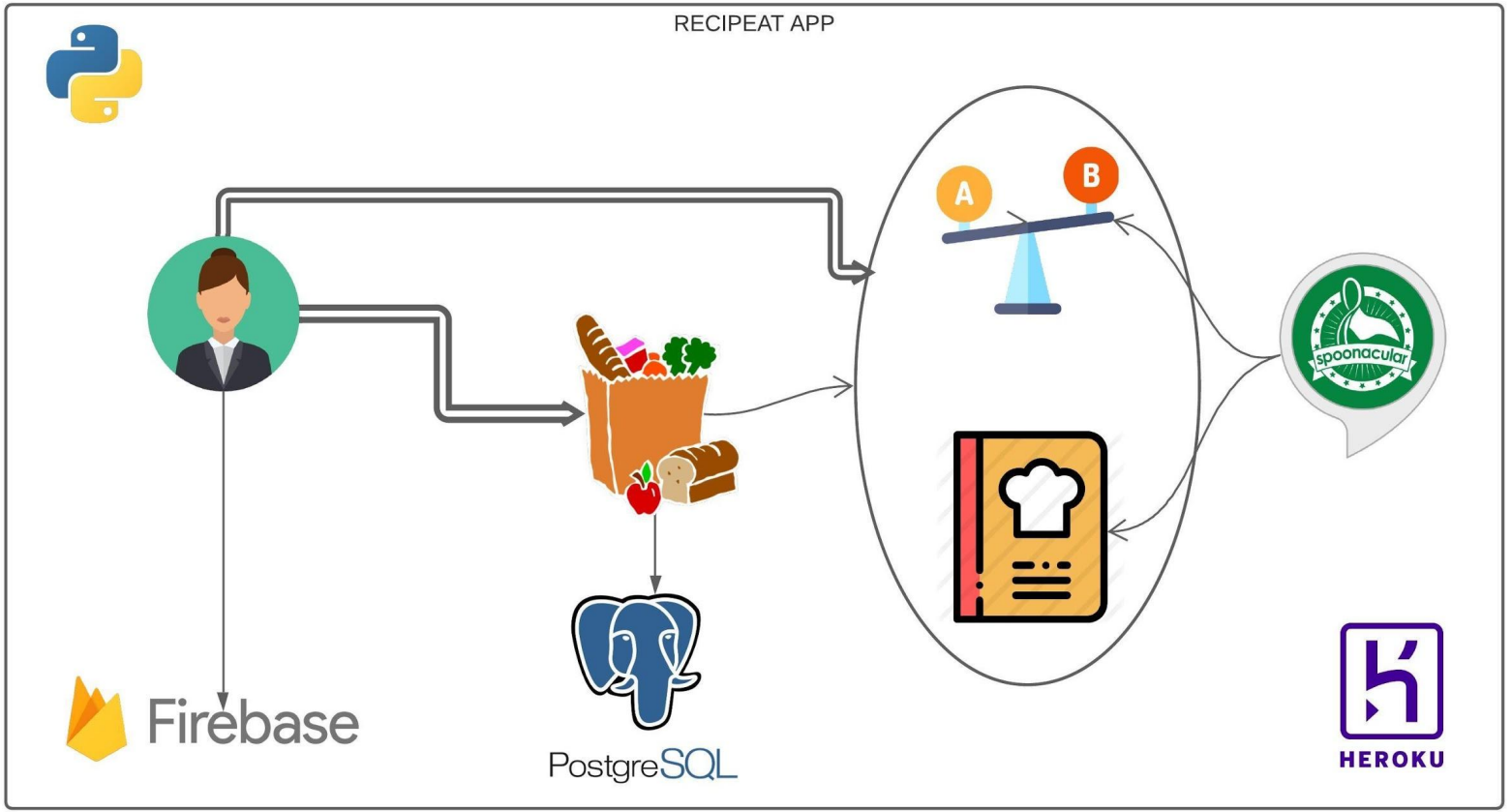
Software Architecture

- ***Web Development***
  - HTML, CSS, Bootstrap
  - Flask Framework

- ***Database***
  - PostgreSQL, Firebase
  - User and Data management

Demo

Design



Project Structure

Refer to GitHub README.md

Lessons Learned

- Documenting scope as much as possible to avoid scope creep
- Integrating Travis earlier on
- Using Django
- Make sure code is Pep8 compliant as you code
- Enabling Test Driven Development is tough when the application keeps changing and there are more dependencies.

Future Work

- Add in logic to recommend recipes based on user ratings of past recipes - building a recommendation system.
- Allowing users to save recipes they like to their profile
- Saving user preferences such as food intolerances
- Creating functionality for “guest” users that don’t want to create a profile
- Making visual comparator more interactive
- Making our application highly scalable and highly available
- Developing a Meal Planner

Thank you!