CSE 454 Project Proposal: Restaurant Oracle

Team Members

Coral Peterson (coralp@cs, GitHub account: coralp)
Laure Thompson (laurejt@cs, GitHub account: laurejt)
Henry Baba-Weiss (htw@cs, GitHub account: htbw)

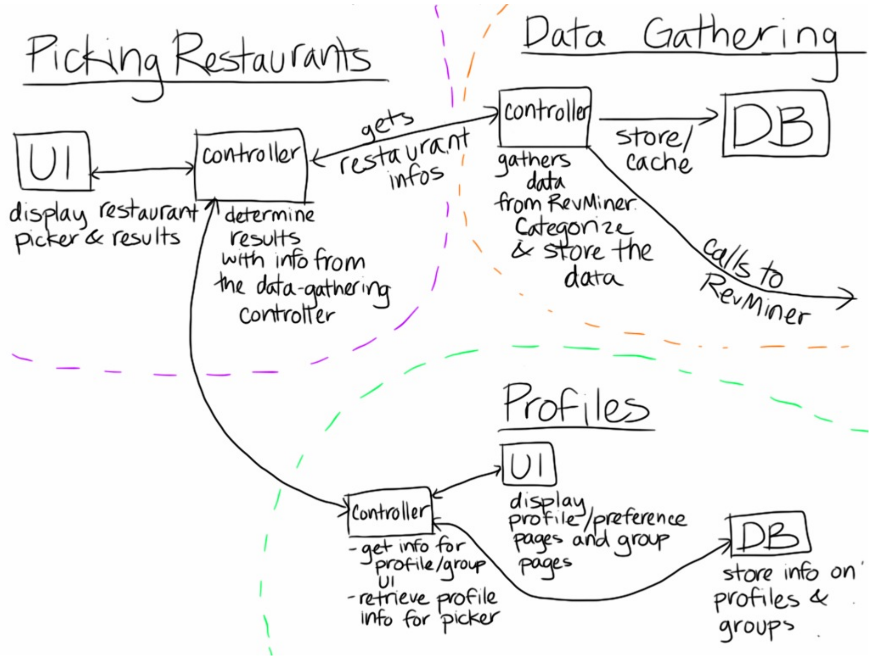
Proposal

Choosing a restaurant is often a difficult problem for groups as well as individuals. Many factors, such as food preferences, location, and price, must be considered when looking for a place to eat. This problem only gets worse as the size of the group grows. Restaurant Oracle will solve this problem by providing restaurant suggestions based on each individual's dining preferences. These suggestions are built off of profiles which indicate each person's likes and dislikes, as well as current information (such as location, time, dining cost, etc.).

Artifacts

- Front-end interfaces (represented by the UI in the diagram):
  - An Android phone app package
  - A web-only interface made in HTML5
- Back-end web services (controllers and databases in the diagram)
- Source code for front-end interfaces and back-end web services
- End-user documentation

Our general system looks like the following:


Picking Restaurants
UI
controller
display restaurant picker & results
determine results with info from the data-gathering controller
gets restaurant infos
Data Gathering
controller
store/ cache
DB
gathers data from RevMiner. Categorize & store the data
calls to RevMiner
Profiles
UI
Controller
display profile/preference pages and group pages
DB
- get info for profile/group UI
- retrieve profile info for picker
store info on profiles & groups

The profile part of our program will have its own MVC components, and will handle creating, modifying and saving profiles and groups. A data gathering service will make calls to RevMiner and organize and store the resulting information in a database. Picking restaurants will have its own controller and UI. This controller will determine the list of restaurants to pick, and the UI will display the restaurant-picking part of the app and the results to the user. The restaurant-picker controller will make calls to the data-gathering and profile controllers to get relevant information.

Technologies

We plan to use HTML5 with jQuery for the front-end development, in addition to our Java-based Android front-end. For the controllers, we plan to use PHP with the standard language features. Our databases will use MySQL. We plan to use AWS for our hosting unless unexpected costs or difficulties occur. If this happens, we will probably put our code on Attu or another to-be-determined server. We plan to gather data offline from RevMiner, filter it for data relevant to our purposes, and store it in a MySQL database. Our UI will be fairly simple, and restaurants will be in a list or on a map.

Roles

We will roughly divide the work as follows, but we anticipate this will change as the project goes on and we discover who enjoys which parts of the project most.

Coral	Henry	Laure
- Skeleton architecture: make hello world pages for profiles - Able to create new profiles and log in / out - UI for the profile and group pages - Profiles work (all sections filled out and saved) - Restaurant algorithm	- Skeleton architecture: set up server/backend infrastructure - UI for the results page, pick a way to display restaurant information - Persistent groups work (people are able to join / leave groups, which are saved) - Restaurant algorithm	- Skeleton architecture: make hello world page(s) for restaurant picker - UI for the query page - Temporary groups and other inputs that exist for only one query are collected - Extracting data from RevMiner and storing it in a reasonable way - Restaurant algorithm

Milestones

- *Jan. 26th (milestone 1)*: Have the basic HTML5 skeleton set up. The restaurant evaluator does not need to work, but it should be hooked up to the rest of the app.
- *Feb. 16th (milestone 2)*: Getting restaurants works accounting for distance and/or travel time, and the preferences of one user. Be able to create a profile and groups.
- *Mar. 6th (milestone 3)*: App is finished. Getting restaurants works accounting for smaller, more temporary preferences (like cost, time of the meal, etc.)

Evaluation

We will evaluate the success of our Android app by user feedback. This feedback will be obtained by having the app ask the user about the quality of the suggested restaurants. This user feedback will be gained by distributing our app to our friends and classmates. We would like ⅔ of our searches to end with a satisfactory use experience (i.e, a good restaurant choice).