README: Getting Started:

To seed services (opentable, fandango, seatgeek, etc.), run:

rake db:seed:services

To import restaurants from Open Table, run:

rake db:import:restaurants

Remember to Put facebook key and secret in instagator/.env

Basic Backend Overview

schema can be found in erd.pdf

How Event Creation Works

Creating an Opentable Event

Step 1: Filling out the Info

Controller Action: users#dashboard

View Code: events/\_new.html.erb

What Happens in backend: nothing

Step 2: Submiting Event Info

Controller Action: events#create

View Code: none

What Happens: An event is created with the submitted params,

1 poll is created for the current user. The poll belongs to the event.

Step 3: Choosing Restaurants Choices

Controller Action: services#opentable

View Code: services/opentable.html.erb

What happens: Restaurants are retrieved

Step 4: Submitting Restauraunt Choices

Controller Action: choices#create

View Code: none

What Happens: Data form the selected restaurants are extracted and turned into Active Record Choices. choice#value is the restaurant name, choice#service\_id is the opentable id, choice#add\_info is the restaurant address and so forth

Step 5: Review Event

Controller Action: events#show

View Code: events/show.html.erb

What Happens: Event's activated status attribute is now changed to 'activated' so it can show up on the dashboard. User can now take poll or go back to dashboard

Creating a Simple/Anything Goes Event

Step 1: Filling out the Info

Controller Action: simple/events#new

View Code: simple/events/\_new.html.erb

What Happens: nothing

Step 2: Submiting Event Info

Controller Action: simple/events#create

View Code: none

What Happens: An event along with its choices is created with the submitted params.

Choices are also assigned a question value and a choice\_type value depending on what was filled out in the form. choice\_type is used to distinguish between an opentable choice and a simple/anything goes choice.

How Voting Works

Voting on an Opentable Event

Step 1: Upvote/Downvoting Choices

Controller Action: choices#index

View Code: choices/index

What Happens: Each click on upvote and downvote modifies the 'yes' boolean attribute on the choice. The choice's new yes\_count is then displayed via ajax.

Step 2: Clicking Submit

Controller Action: choices#decide\_vote

View Code: none

What Happens: All votes for the choice's event are tallied up. If the choice with the highest number of 'yes' votes has exceeded the event's 'threshold' attribute. An api call is made to the opentable bot to book the reservation This is done in opentable#reserve

Voting on a Simple/Anything Goes Event

Step 1: Clicking on choices

Controller Action: choices#index

View Code: choices/index

What Happens: This time, clicking on choices does not make ajax calls to the backend. Clicking on choices simply toggles the html classes to make them appear selected. Votes are not saved until user hit's submit.

Step 2: Clicking Submit

Controller Action: polls#vote

View Code: none

What Happens: All selected choices for the poll are submitted at once and updated. No bot action occurs.

How the Opentable Bot Interaction Works

When a poll's choice's 'yes\_count' becomes equal to it's parent event's 'threshold' for the first time, it triggers an api call to the Opentable bot hosted on another web app. The bot uses the choice and the event's info to book a reservation within the time range specified by the instagator when they first created the event.

If successful, the event's 'current\_choice' attribute is updated with the selected choice's 'value' attribute, and the instagator is emailed by Opentable.com. If not successful, the event's 'processing\_choice' attribute is updated with the selected choice's 'value' attribute, and the instagator is emailed a url by which he can make the reservation manually. A separate email is sent out to site staff containing the necessary information to make the reservation manually on behalf of the instagator.

Coffeescript

Coffeescript files are named according to the the controller action they are related to.

Some notes on simple/events\_new.js.coffee to help with any refactoring and changes that might need to be implemented in the future:

The simple events form creates choices using a datepicker and a text choice picker.

The datepicker is actually two datepickers for the purpose of displaying two months. In order for this to function properly, the datepickers need to be synced. This means changing the month on one datepicker, needs to shift the other in the same direction as well. Setting a date on one should set the date on the other as well. This was done by creating callback functions for the changeMonth and changeDay events on both datepickers.

All questions in the form share a single datepicker and text choice picker div. They are refreshed after creating a question. And when a user clicks in to edit a question, the data is read from hidden input fields in the div to populate the text/date choice picker.This gives off the effect that each question has it's own choice picker when it's actually just two that are moving around.

Submitting a question saves the selected choices into the hidden input fields nested in the question divs. Submitting the entire form saves the entered questions into another hidden input field. These fields are parsed by events#create\_simple\_event to create an event as well as it's choices.