htchr

Group Members:

Dima Ivanyuk (divanyuk)
Matt Schallert (mschalle)
Salem Hilal (shilal)

Note: Iterative designs / user studies can be found in the `/docs` folder **Note 2:** The iOS simulator for Mac has known issues with obeying `device-width` and similar `meta` tags, so the video does not 100% accurately reflect the visual performance.

Required Features

1. Javascript
    - public/javascripts/htchr.js for the frontend
    - app.js, routes/*, db/models.js for serverside

2. DOM Manipulation
    - Populating news feed
    - Viewing events
    - Autocomplete
    - htchr.js Lines 170-260

3. jQuery
4. jQuery Mobile
    - Front end minus landing page

5. AJAX : Consume API
    - Google Maps/Places for geolocation
    - Facebook - hook into user profile, find friends, etc.

    - Twilio - send notifications about events

6. AJAX : Provide API
    - app.js has RESTful routes hidden behind facebook auth.

7. node.js
    - Used express throughout, taking advantage of things like express sessions, static directory mappings, and server-side view rendering

8. Server-side DB
    - MongoDB
    - Mongoose
    - db/*; querying in myUtil.js, routes/*

9. EJS templates
    - See views/*
    - ~ lines 30, 190, 230, 250
    - Also some client-side underscore templating going on

10. PhoneGap

General Description of Features

1. Event Feed
    - Provide a chronological overview of events as they are created/happening
    - Updated on the fly

2. View Event
    - Separate page exists for each event
    - Default image for the event is set to the Google Places image associated with the event's location

3. Facebook-Integrated
    - Login with Facebook (Passport)
    - Creates corresponding events on Facebook for each event made in

htchr
    - Events viewable by default only to Facebook Friends

4. Event Search
    - Attempt to find a similar event within the user's friendbase
    - Attempts to find nearby events if an exact match isn't found.
    - Uses event name by default, but optionally include time/date range(s)

5. Create Event
    - Alternatively, create an event if one cannot be found
    - Control privacy of your event and who to initially invite

6. Autocomplete
    - Autocompletes places using Google Places API, biased by proximity of place to user's physical location
    - Autocomplete friends names when inviting users to events

7. Notifications
    - Allow users to give us their phone numbers and be notified of event invites and upcoming events

8. Event Recommendation
    - Recommendation system attempts to pair up users and events
    - Events are recommended to users in the event feed
    - Upon event creation, htchr suggests friends to invite

Something Extra

As part of our project, we also looked into how modern web applications are hosted at real-world scale. As such, we learned how to structure our infastructure and our application to potentially prepare to take our site to a larger public audience some day. We set up our own infastructure on Amazon Web Services. We have a single mongoDB database server, and currently one express application server. In front of our application server is an Elastic Load Balancer. This means that if we ever launch and have a rush of users, we can replicate our

most recent snapshot of our application server (~4 minutes), pull the most recent code to the new server (~1 minute), and put the server behind the load balancer (~3 minutes for changes to propegate across network). Thus in under 10 minutes, we can scale our overall infastructure to handle twice as many users, and thus won't be bottlenecked until we hit heavy load on our DB server (at which point we would look into data sharding techniques to deal with the new load).

This was an unanticipated learning experience we gained in our project, and it was really interesting to see how real-world applications are created and served.