CIS 450/550 Tripster

Final Report

The Team:

* Quanze (Jim) Chen [cquanze]
* Leonard Loo [leoloo]
* Chenyang (Ray) Lei [chenylei]
* Darren Yin [darreny]

Introduction:

The CIS 450/550 Final Project Tripster was constructed using Node.js and a MySQL database

running on an Amazon EC2 instance. Along with Node.js we utilized Express, Jade, Bootstrap, Jquery, S3, and Redis for the project.

Project Goals:

* Users
  + Register new users
  + Existing users able to login
  + Send/accept/decline friend requests
* Trips
  + Creation of new trips
  + Invite/Accept trip requests from friends
  + Rate and comment trips
* Sharable Media
  + Add photo albums to trips with privacy settings
  + Rate and comment media
* Newsfeed
  + Photos and trips viewable by existing friends
* Recommendations
  + Recommend locations/friends based on previous trips
* Other
  + Search for users, trips, and locations
  + Caching photos and videos (w/ Amazon S3)
  + Caching login hotdata (w/ Redis)
  + Notifications
  + Checklists

Extra Credit Claimed:

* Checklists (usable as expense tracker)
* Notifications

Basic Architecture:

The basic architecture of our site is Node.js running on the Express framework. Redis is used to cache login hotdata and S3 is used to cache photos and videos. Frontend features use Bootstrap, Jquery, and Jade.

Shareables Implementation (shareable media)

To allow for shareable media we created an abstract object called a shareable which had a shareable id and a data type (to signify if it was a photo, album, url, or video). We kept the links for the actual media in their own tables with corresponding attributes for each media type.

We chose to do this to prevent one large table with an excessive number of null cells (due to different media having different attributes). We believe that having 1 unique shareable id and then joining the shareables table with any of the media tables was a better solution to obtain information about a specific media type. These shareables must all be added to an album to exist (with an album being contained by itself).

These shareables can then also be shared to other users or to trips (with a privacy setting when shared to trips). The only two privacy settings that exist are public and private. While private only friends and the user can view the shareable while public means that anyone can view the shareable.

Recommendation Implementation

Our recommendation system recommends both friends and trips. It recommends the friends of your friends who are currently not your friends. For trips, it recommends trips that you haven't been to, that at least one friend is currently a member of.

Search Implementation

Search is implemented by doing a LIKE match in our MySQL database. It then returns any users, trips, and locations that match the search string or contain a substring that is equal to the search string.

Caching Implementation

Redis for caching login hot-data, i.e. user data is cached when people log on. And we are caching photos and videos in S3.

Key Technical Challenges:

* Importing Data:
  + The data given was bad leading to issues with importing data
  + Led to manually importing good portions of data by hand
* Tweaking database structure:
  + The structure of the relationships in our database changed over development
  + Difficult to create a schema that works first time around
  + Difficult to also maintain most updated schema when modifications are made

Performance Evaluation:

Potential Future Extensions:

* Integration with Yelp data
* Usage of Bing search
* Expense tracker
* Importing user info from FB/G+