CIS 450/550 Tripster

Final Report

The Team:

* Quanze (Jim) Chen
* Leonard Loo
* Chenyang (Ray) Lei
* Darren Yin

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, 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 and locations
  + Caching Implementation (Amazon S3)
  + Notifications

Basic Architecture:

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

NO IDEA RIGHT NOW

Search Implementation

JIM CAN WRITE

Caching Implementation

NO IDEA RIGHT NOW

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+