# Course Project

# Outline

- Take data stored in flat text files and create a database
  - Normalize, create relations, import into DB
  - Create triggers/stored procedures as necessary to enforce constraints
- Prepare SQL queries for common scenarios
  - Inserting/updating/deleting/selects/joins
- Use indexes (and perhaps denormalization) to optimize query performance

# Project

- Social networking site like Foursquare or Yelp
  - Users are friends with other users
  - Users follow other users
  - Users check in at locations
    - possibly with friends
  - Users review locations/businesses
  - Users comment on friends' reviews
  - Users rate reviews

- Work alone, in pairs, or in a group of 3

# Data

- I started with four large files:
  - User data (~200,000 records)
  - Friend/following data (~2 million records)
  - Location data (~1.3 million records)
  - Check-in data (~6.4 million records)

- I have cut them down to:
  - User data (~10,000 records)
  - Friend/following data (~81,000 records)
  - Location data (~221,000 records)
  - Check-in data (~564,000 records)

# User Actions

- A user should be able to:
  - create a new account
  - log in to an existing account
    - (password authentication NOT required)
  - follow/friend other users
    - and unfollow/unfriend
  - view friend/followee checkins
  - check in at a location (possibly with other users)
    - if location is not already in DB, user should be able to provide details about it
  - review a location
    - delete/edit own reviews
  - rate other users' reviews
  - comment on friends' reviews

# Project Phases

- Three checkpoints before final submission:
  - 5/2: have relations created, data normalized and imported
  - 5/9: write queries to handle common tasks; verify they work; empirically evaluate efficiency
  - 5/16: optimize query performance; compare performance before/after optimization

5/25: final submission, incorporating things from all three checkpoints

Only the final submission will be graded. Submit checkpoints if you want feedback from me (otherwise they are optional)

# First Checkpoint

- 5/2: Create relations, **normalize data**, import
  - Check-in data has a lot of redundancy
  - Identify functional dependencies that capture redundancy
  - My suggestion: write a program to decompose file into several new files suitable for importing into BCNF relations