Datastore models

1) User
email : string
password : string
date_created : date
first_name : string
last_name : string
name : string

2) Circle
date_created : date
description : string
name : string
email : string

3) Question
date_created : date
description : string
circles : list of strings
email : string
access_list : list of emails
name : string
location : string

4) Notification
date_created : date
data : string
is_read : string
email : string
creator_email : string
creator_name : string
type : integer

5) Answer
description : string
date_created : date
email : string
upvote_count : integer
question_id : integer
name : string

6) Vote
email : string
answer_ids : list of integers
name : string

7) Favorite
question_ids : list of integers

8) Contact
email : string
date_created : date
circles : list of strings
user_email : string
name : string

URL's to be implemented

All responses from server will be converted to JSON format.

/home

POST -> (email, question_id, favorites)
returns: jinga template rendering the questions and answers (setting question_id will return only the answers for that particular, and setting favorites returns only list of questions favorited by users)

/create_user

POST -> (email, password, first_name, last_name)
returns: "Failure" or redirects to login page

/create_circle

POST -> circle_name, circle_description, email
returns "Success" or "Failure"

/post_question

POST -> comma separated circle names, email, name_of_user, location, question_description
returns "Failure" or question identifier in case of success

/post_answer
POST-> question_id, email, answer_description, user_name
returns “Failure” or answer identifier in case of success

/create_contact
POST-> comma separated circle list, name of contact, email of contact, email of user
returns “Success” or “Failure”

/view_contacts
POST -> email
returns jinga templated rendered with contacts

/mark_vote
POST -> answer id, question id, email, name
returns “Failure” or upvote count
When a user wants to see a particular question, the id is posted from client side and the data is retrieved from the server and posted separately in a web page.

/mark_favorite
POST -> question_id, email
returns “Failure” or list of ids of favorite questions.

Miscellaneous

We’ve used google’s channeling api for communicating the server with the clients and used server side Jinga templating for rendering HTML pages.

Requirements

Name	Our choice
External Javascript library	validate.js (for form validation)
Google App Engine Advanced Feature	Channeling API
HTML5	Geolocation
Templating language	Jinga

Data Store Model Diagram:

The entity models are arranged using links so that it becomes easier to visualize the entity groups. For a given model, all the models in the subtree with the table as root are its descendants. As discussed during the course of the project, we removed all the soft dependencies, keeping performance in mind. Strong consistency is required only during the case of posting answers to the questions and hence this link is preserved.