*Srikanth Srungarapu [srungar2]*

*Vishnu Priya Muthukrishnan [muthukr2]*

**Technical Design Documentation**

**Quora+**

**Datastore models**

* **Users Object**

First Name: String

Last Name: String

Email: String

Password: String

Date Created: Date

* **Questions Object**

id: Integer (generated randomly)

Date Created: Date

Description: String

* **Answers Object**

id: Integer (generated randomly)

Description: String

Email: String

* **Upvotes Object**

id: Integer (generated randomly)

Email: String

Date Created: Date

* **Favorites Object**

id: Integer (generated randomly)

Date Created: Date

Email: String

* **Share Object**

id: Integer (generated randomly)

Circle Name: String

* **Notifications Object**

id: Integer (generated randomly)

Description: String

Date Created: Date

* **Circles Object**

id: Integer (generated randomly)

Name: String

Description: String

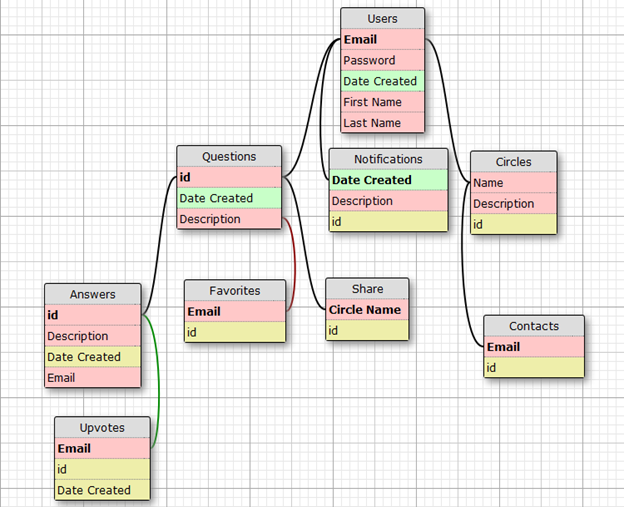
* **Contacts Object**

id: Integer (generated randomly)

Email: String

**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.



**URL’s to be implemented**

All responses from server will be converted to JSON format.

**/main\_page**

GET -> (no parameters)

returns: [Question1, Question 2, Question 3, ..., Question N]

get a list of the questions including the ones that was shared with this particular user.

**/user\_id/vote**

POST-> answer\_id, type in [“up”, “down”]

returns “success” or “failure”

**/user\_id/favorite**

POST -> question\_id, type in [“add”, “delete”]

returns “success” or “failure”

**/question/id/display**

GET -> (no parameters)

returns: [{answer: “description”, count: “upvoters\_count”, upvoters:[“voter1”, “voter2”]}....]

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.

**/user\_id/display\_contact**

GET -> (no parameters)

returns [{name : “contact1”, email: “contact1@gmail.com”, circles:[“friends”, geeks]}, .....]

The set of contact details are listed from the server.

**/user\_id/create\_contact**

POST -> Name of new contact, List of Circle names, Email of new contact.

returns “success” or “failure”

Having posted these details, a new contact is created on the server.

**/user\_id/create\_circle**

POST -> Name of the new circle.

With a given circle name, a unique id is generated on the server side and this circle is added to the circle list.

**/user\_id/post\_question**

POST -> circle\_id, question

Creates a new question. The new\_question parameter accepts a question object.

**/user\_id/post\_answer**

POST-> question\_id, answer

**Miscellaneous**

We’re planning to use web push notifications api for pushing updates to the clients and clients will use normal GET/POST methods for sending requests. We’re planning to memcache the answers object and questions object as multiple users can access that information and given that we’re not providing edit functionality for answers, caching can provide good performance.

**Requirements**

|  |  |
| --- | --- |
| **Name** | **Our choice** |
| External Javascript library | YUI |
| Google App Engine Advanced Feature | Task Queue |
| HTML  5 | Web Push, Update Stream |
| Templating language | Dust.js |