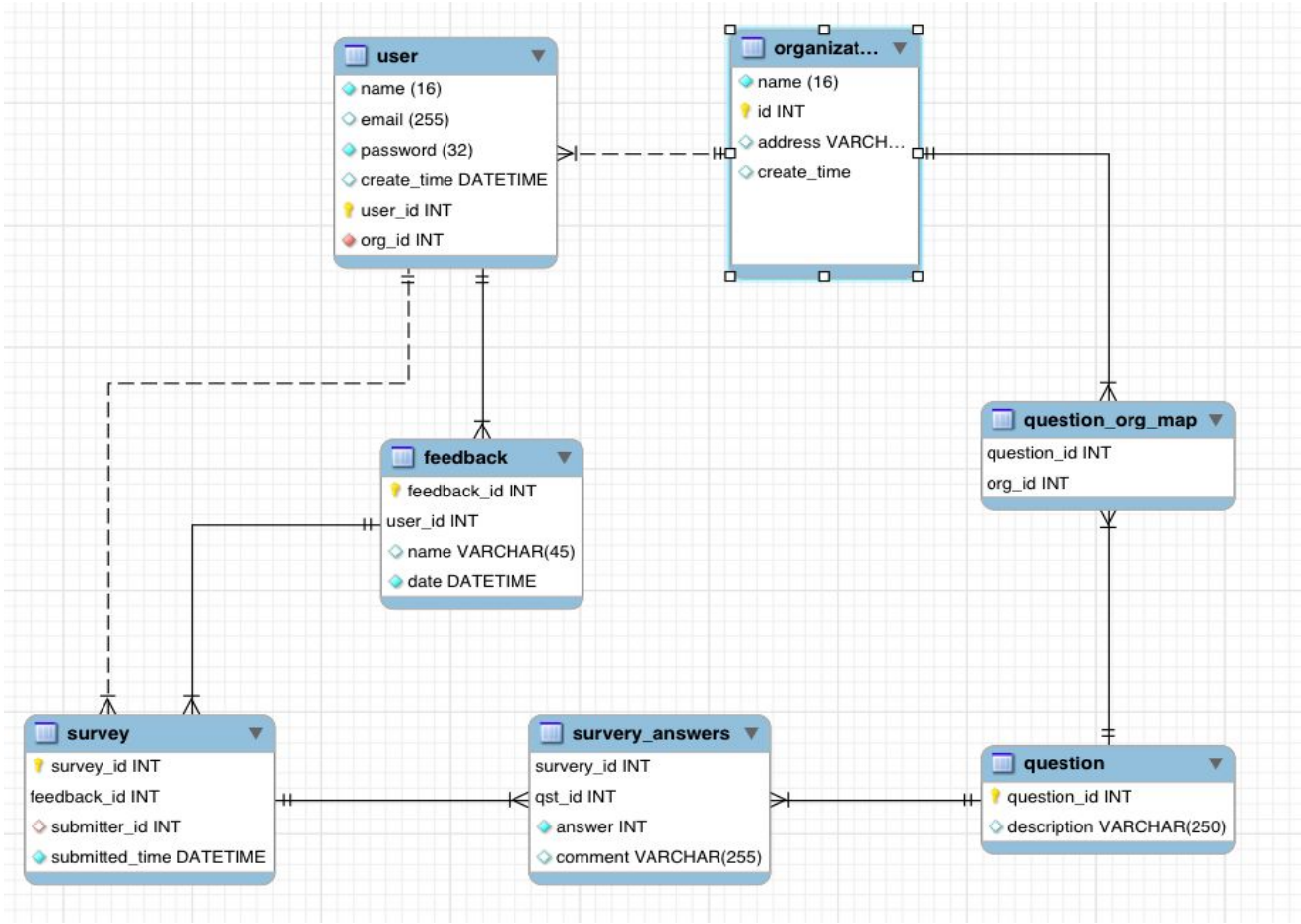
INTRODUCTION:

A data model and basic UI for a simple peer to peer application is being constructed below. The main functionality of this application is that a user (recipient) can request feedbacks from other users(submitters) and also write feedbacks.

ER-DIAGRAM:



Explanation:

From the given requirements, the main entities are Organization, User, Feedback, Survey and Questions. And the following are the relationship between those entities.

Organization to User - 1 : n relationship
User to Feedback - 1 : n relationship (Here user is recipient)
Feedback to Survey - 1 : n relationship
User to Survey - 1 : n relationship (Here user is submitter)
Question to Organization - many : many relationship
Survey to Question - many : many relationship

Rough Steps with respect to the above model

- When user creates a feedback and selects the submitters (Say 5)

- one record is added to the “feedback” table with that user as recipient.
- 5 records are added to the “survey” table one for each submitter.
- 5 * (number of qsts) records are added in “survey_answers” table

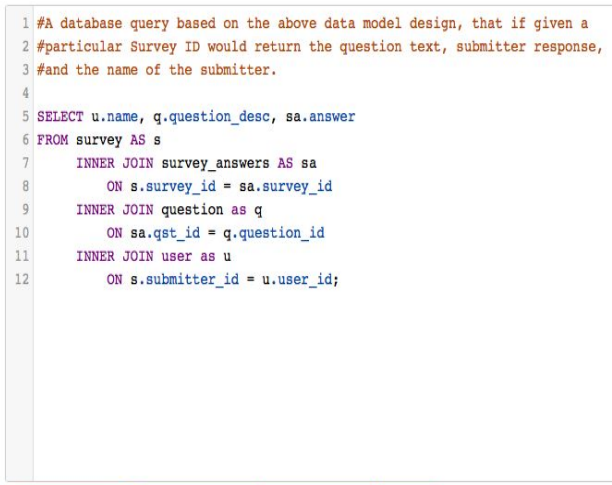
- When the submitter submits the record
  - The record of “survey_id” and “question_id” in “survey_answers” table gets updated with the answers

- When the recipient views the response
  - A list of all his/her feedbacks are displayed
  - For every feedback, all the survey details, questions and it’s corresponding answers are queried from database and displayed.

DATABASE QUERY:

The following is the query based on the above data model design, that if given a particular Survey ID would return the question text, submitter response, and the name of the submitter. (link to the query -> http://sqlfiddle.com/#!9/ba120c/3/0)

```
SELECT u.name, q.question_desc, sa.answer
FROM survey AS s
    INNER JOIN survey_answers AS sa
        ON s.survey_id = sa.survey_id
    INNER JOIN question as q
        ON sa.qst_id = q.question_id
    INNER JOIN user as u
        ON s.submitter_id = u.user_id;
```



name	question_desc	answer
Mary	Rate my timeliness	5
Mary	Rate my overall work	4

Record Count: 2; Execution Time: 0ms View Execution Plan link

ALGORITHM:

Assumptions:

1. All the questions have equal weightage
2. Questions are rated on a scale of 0 to 5 with 0.5 incrementals.
3. Based on above point, every user can get anywhere between 0 to 5 as average score.
4. Every survey has equal number of questions (Made this assumption as organization choose the question not the user)

Pseudo Code:

```
function calculateFeedbackAvg (int feedbackId) {
    int[] sId = getSurveyIds(feedbackId);      //get list of survey ids for the given feedback
    int avg = 0;
    for  each id in sID {
       avg = avg + calculateSurveyAvg(id);     //calculate the average for each survey
    }
    return avg/sid.size();

function getSurveyIds(int id) {
    int[] ids = executeCommand( “SELECT s.survey_id FROM feedback as f INNER JOIN survey
                        as s on f.feedback_id = s.feedback_id WHERE f.feedback_id = %d”, id);
    return ids;

function calculateSurveyAvg(int sId) {
     int[] answers =  executeCommand( “SELECT s.answer FROM survey as s, survey_answers
                                        as sa WHERE s.survey_id = %d”, sId);
     int sum = 0;
     for each ans in answers {
        sum = sum + ans;
     }
     return sum/answers.size();
```

```
def calculateFeedbackAvg (int feedbackId):
    int[] sId = getSurveyIds(feedbackId)    #get list of survey ids for the given feedback
    int avg = 0
    for id in sID :
     avg = avg + calculateSurveyAvg(id)     #calculate the average for each survey

    return avg/len(sid)

def getSurveyIds(int id):
   int[] ids = executeCommand("SELECT s.survey_id FROM feedback AS f INNER JOIN survey  AS s ON f.
       feedback_id = s.feedback_id WHERE f.feedback_id = %d", id)
   return ids

def calculateSurveyAvg(int sId):
    int[] answers =  executeCommand("SELECT s.answer FROM survey AS s, survey_answers AS sa WHERE s.
        survey_id = %d", sId)
    int sum = 0
    for ans in answers:
         sum = sum + ans

    return sum/len(answers)
```

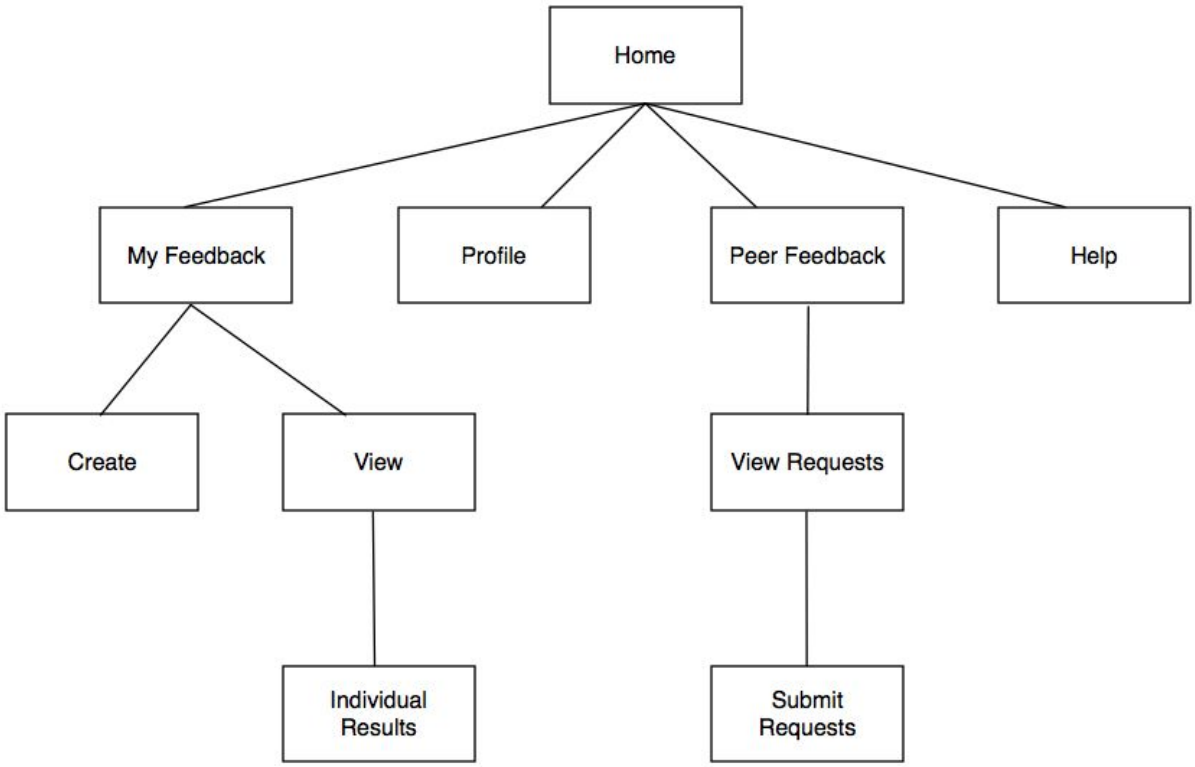
The time complexity for the above algorithm takes n(no of qsts * no of surveys). In the above pseudo code, for the basic feedback average, I am calculating average of every survey (total score / no of qsts) and then calculating the average of surveys per feedback.

WIREFRAME:

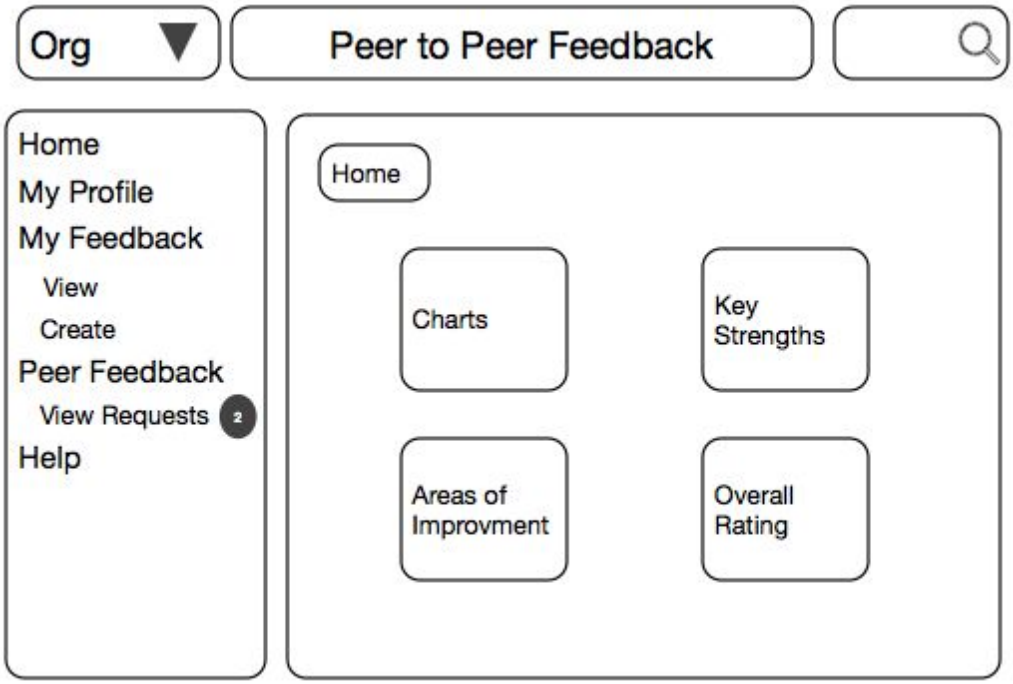
Assumptions:

1. Web Application (With few changes to web UI, the same can be accommodated for Mobile Version too)
2. Every User can be a recipient or submitter within an organization (so using single portal for both the roles)

UI Flow Chart:

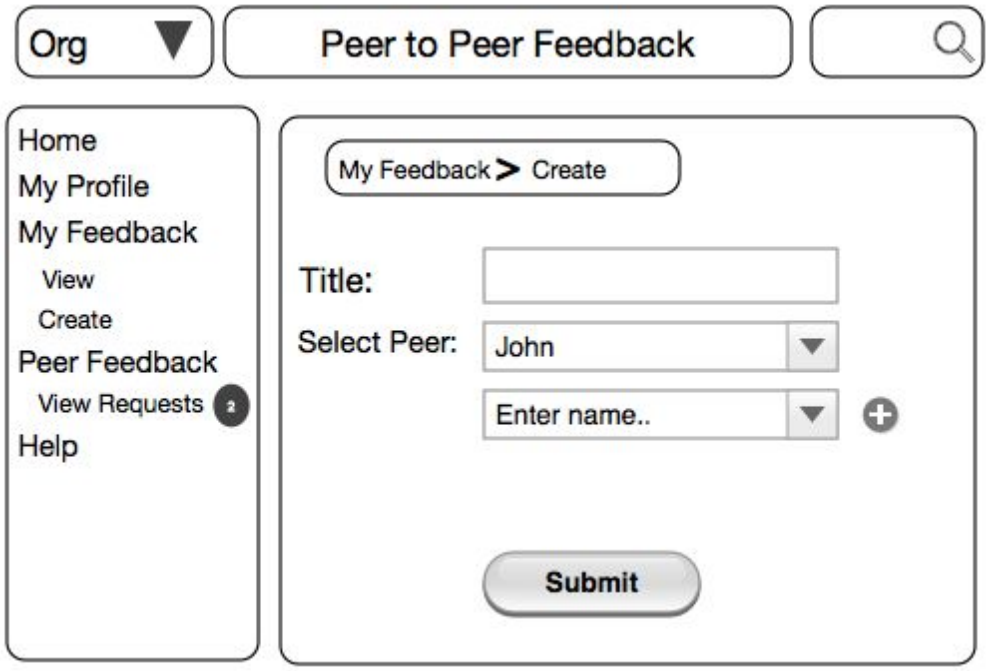


Home Page:



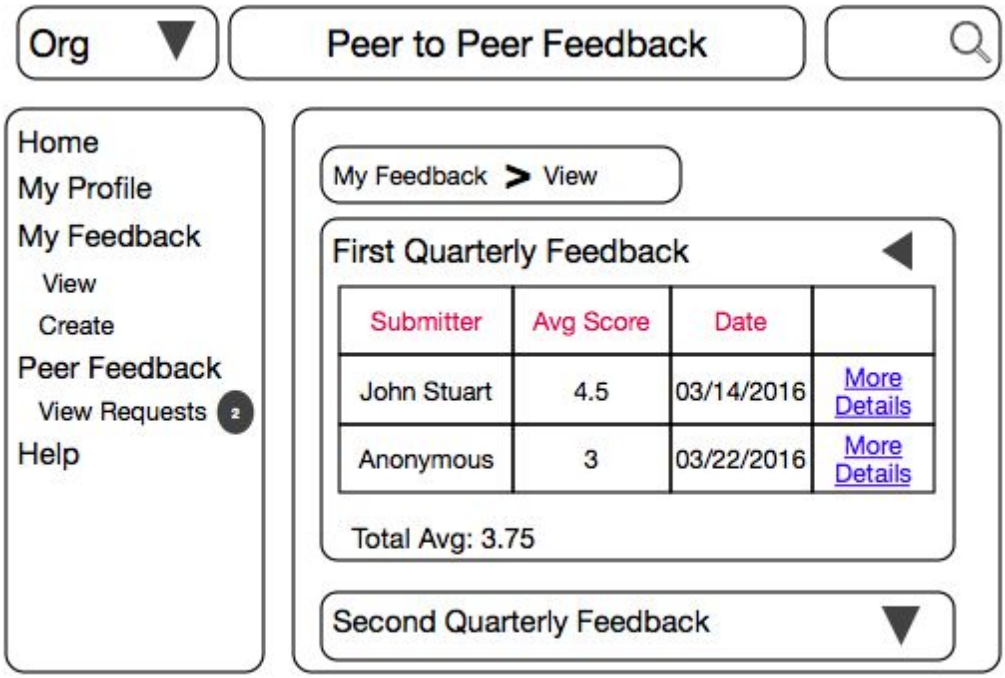
This page is the landing page of the application and serves as a dashboard. It displays the charts about the statistics of his feedback response, summary of key strengths etc. The Header and the navigation (left pane) shown on this screen will be part of every page of this application allowing easy access for the user. In header, when you click on the downward arrow of the “org”, you can see the details of your Organization, sign in or sign out button.

My Feedback -> Create:



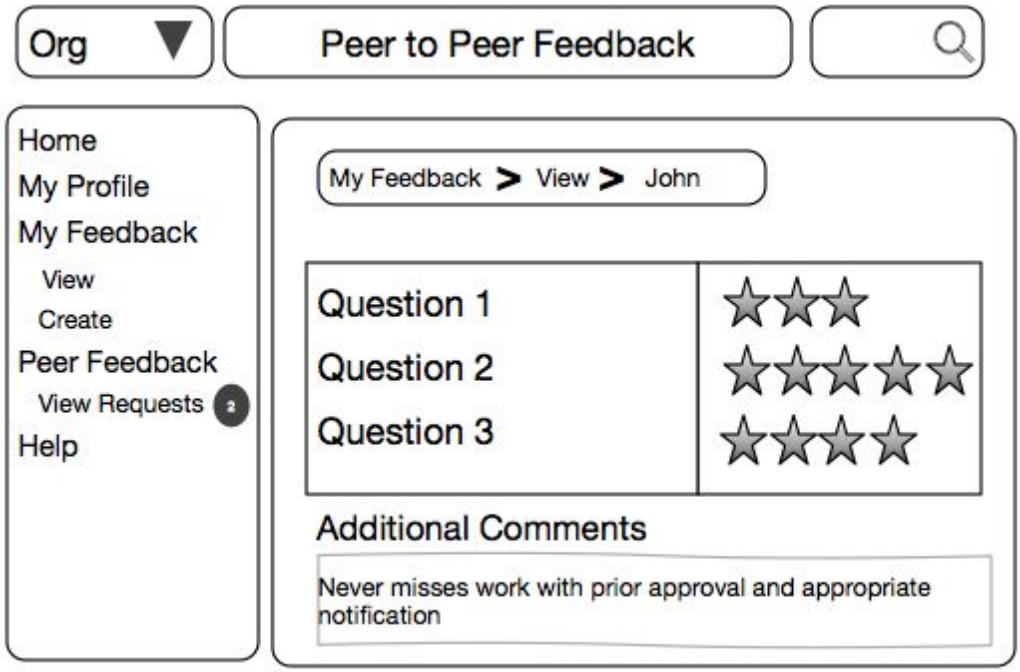
My Feedback is where you can create or view your feedbacks (where your role is recipient). In this create page, you can provide a name for your feedback (optional) and add the number of submitters you would like to request. With the help of plus button, you can add upto 6 submitters for this feedback request. WIth the drop down box, you will get all the other users from your organization. When you submit the request, the submitters will be notified about your request.

My Feedback -> View:



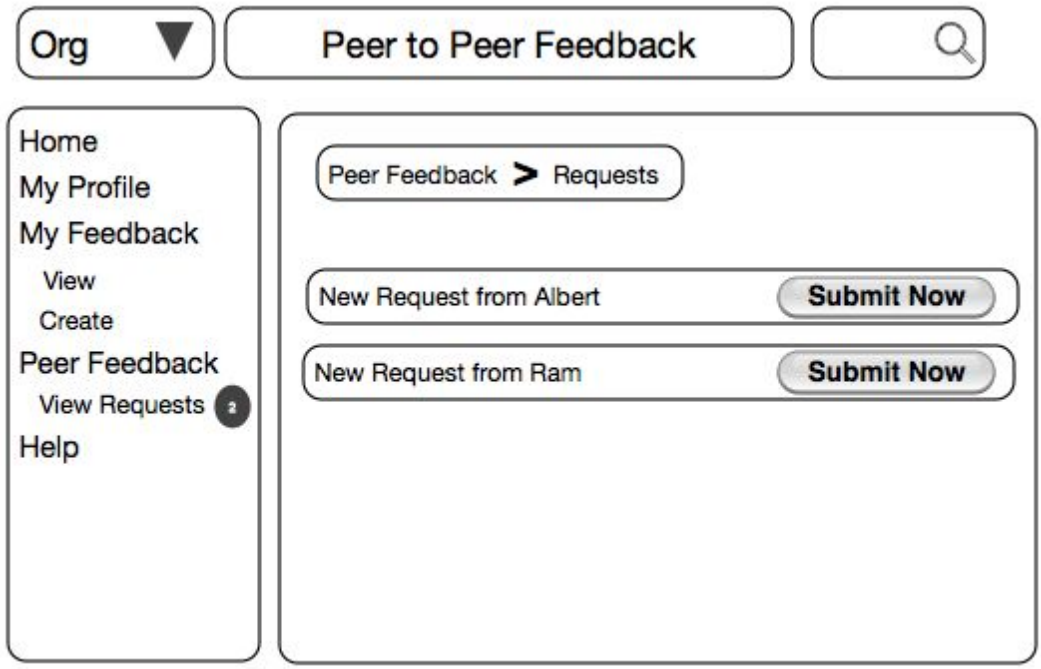
In this view page, the list of feedbacks the user created so far can be seen. By clicking each feedback, it expands to give a list of submitters, their average score and option to view the individual answers to the questions. Also the total average of all the submitters is also visible. If there is any new results from the submitter, you can see a notification (new) indicating that something has been recently submitted.

My Feedback -> Individual Response:



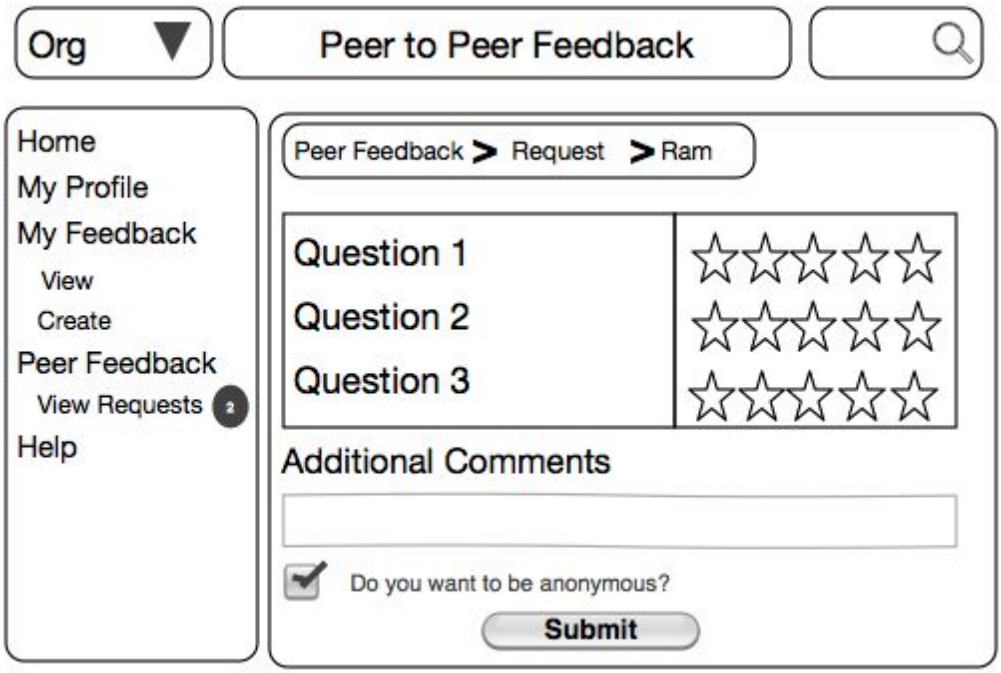
When a user clicks the 'More Details' link in the View Page, the user will be redirected to the above screen. It displays the individual questions and the corresponding answers. The comments are also visible if the submitters provided them with one. Again, here the breadcrumb says which submitter's (if not anonymous) results you are viewing.

Peer Feedback -> View Requests:



Peer Feedback is where we can see requests for feedbacks that we receive from other users. Whenever there are new requests, you will see the number of requests in the navigation bar. Here you will see the list of requests and “submit now” button.

Peer Feedback -> View -> Individual Request:



The user can enter his/her response and press submit button. The user has the option to be anonymous. If he/she choses to be anonymous, the submitter id won't be stored in the survey table.

Help and My Profile Page:

Help page will have a small video about how Peer to peer feedback works and My Profile page will give the user the option to edit or update their details.

SUMMARY

A data model has been constructed according to the given set of requirements. And by making few assumptions, a basic user flow and simple average scoring systems are implemented above.

All these documents including the actual ERD model, the Omnigraffle sketch and the database code can be found in this github page (https://github.com/KavithaVishwanathan/Feedback).