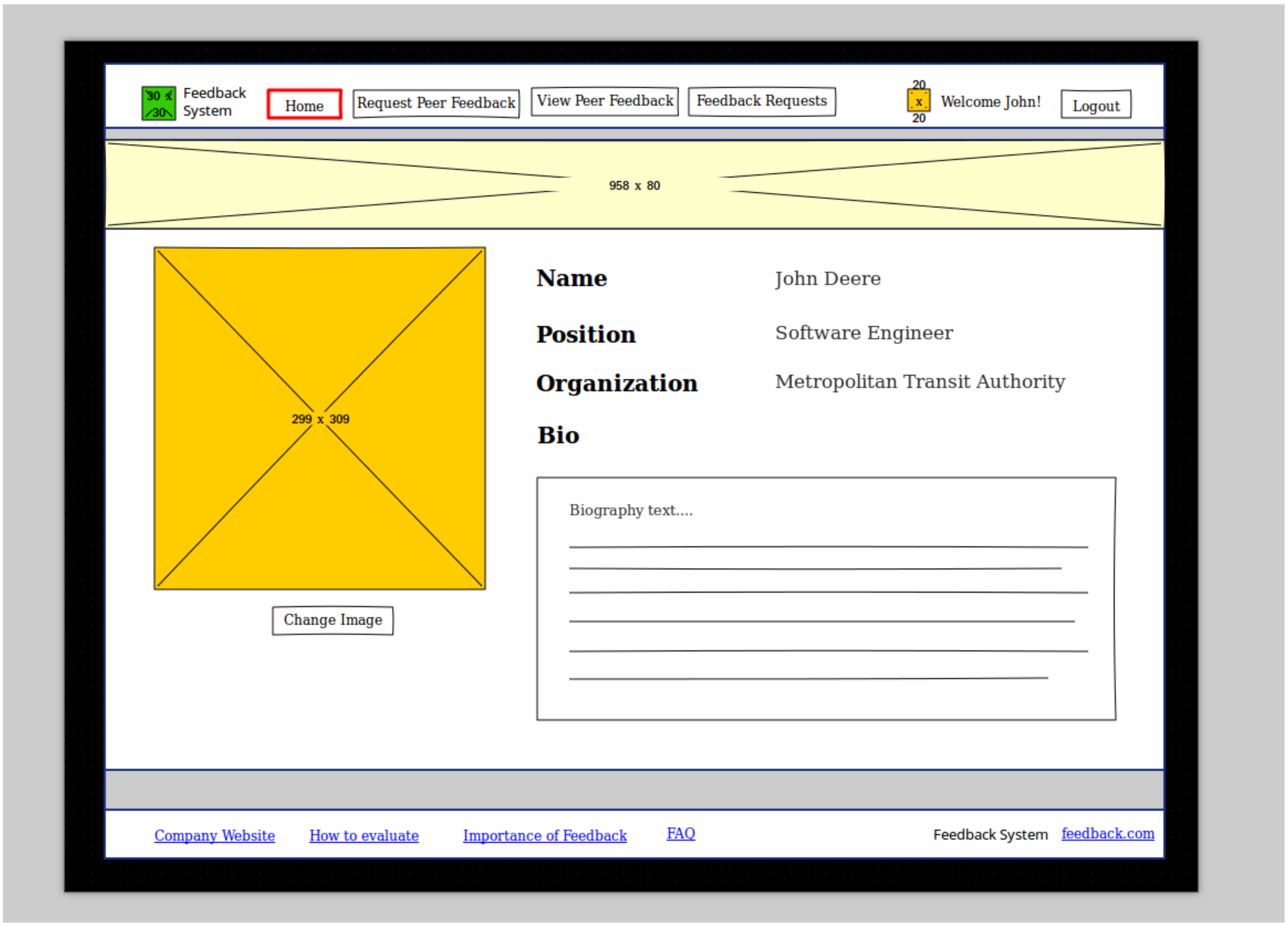
360 Degree Feedback System

Wire Frame Design

Notes:

* Solid red lines around buttons suggest that this is the current open tab.
* Dotted red lines around buttons suggest that this will be the next button clicked.
* Green box on nav bar suggests an image, either for the company of the employee or the logo of the creator of the system.
* Orange box on the nav bar suggests an image of the employee. Could be an avatar, or a picture of the employee.
* Boxes that have a gray background suggest that a modal was opened.



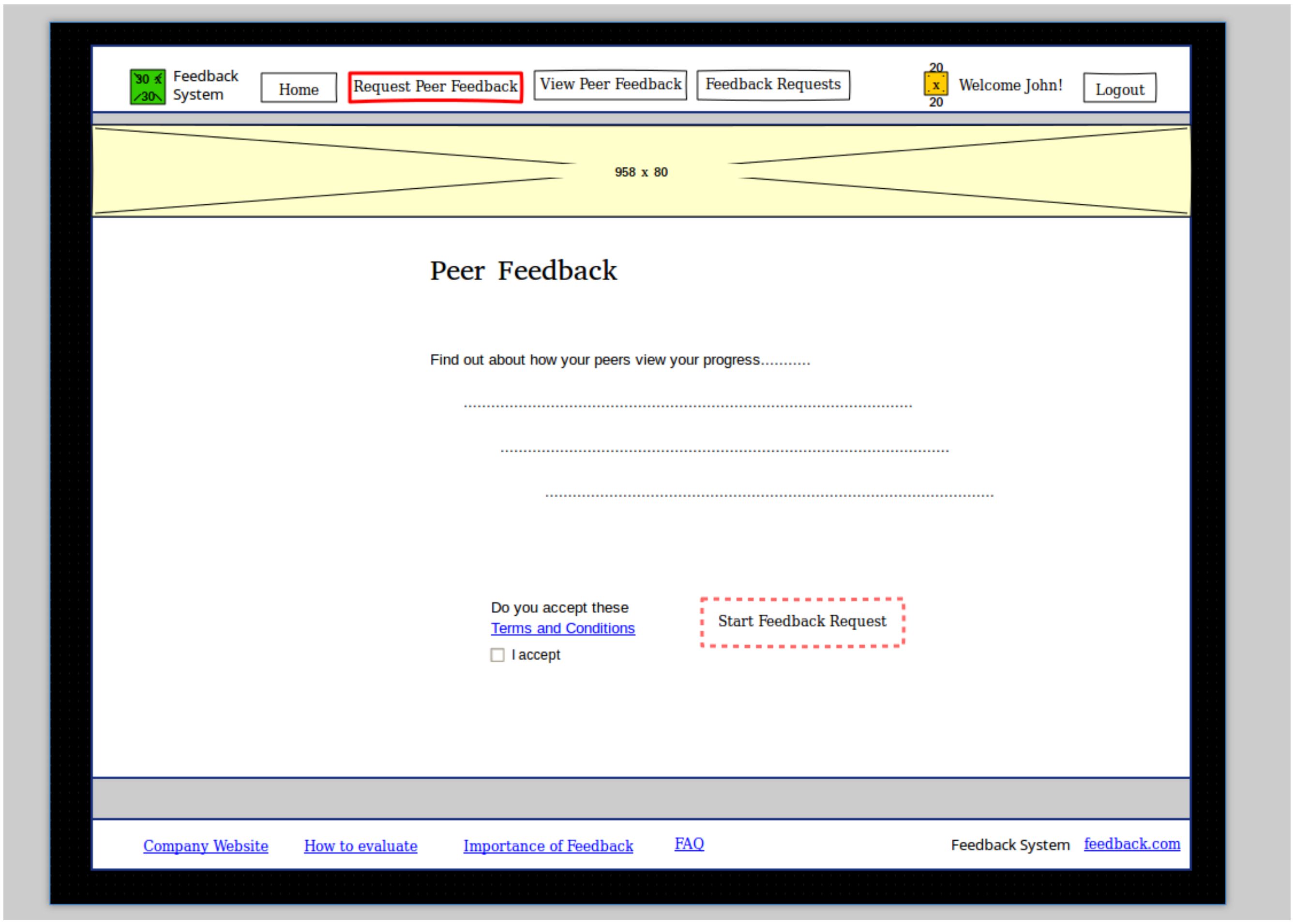
Name John Deere

Position Software Engineer

Organization Metropolitan Transit Authority

Bio

Biography text....



Peer Feedback

Find out about how your peers view your progress...........

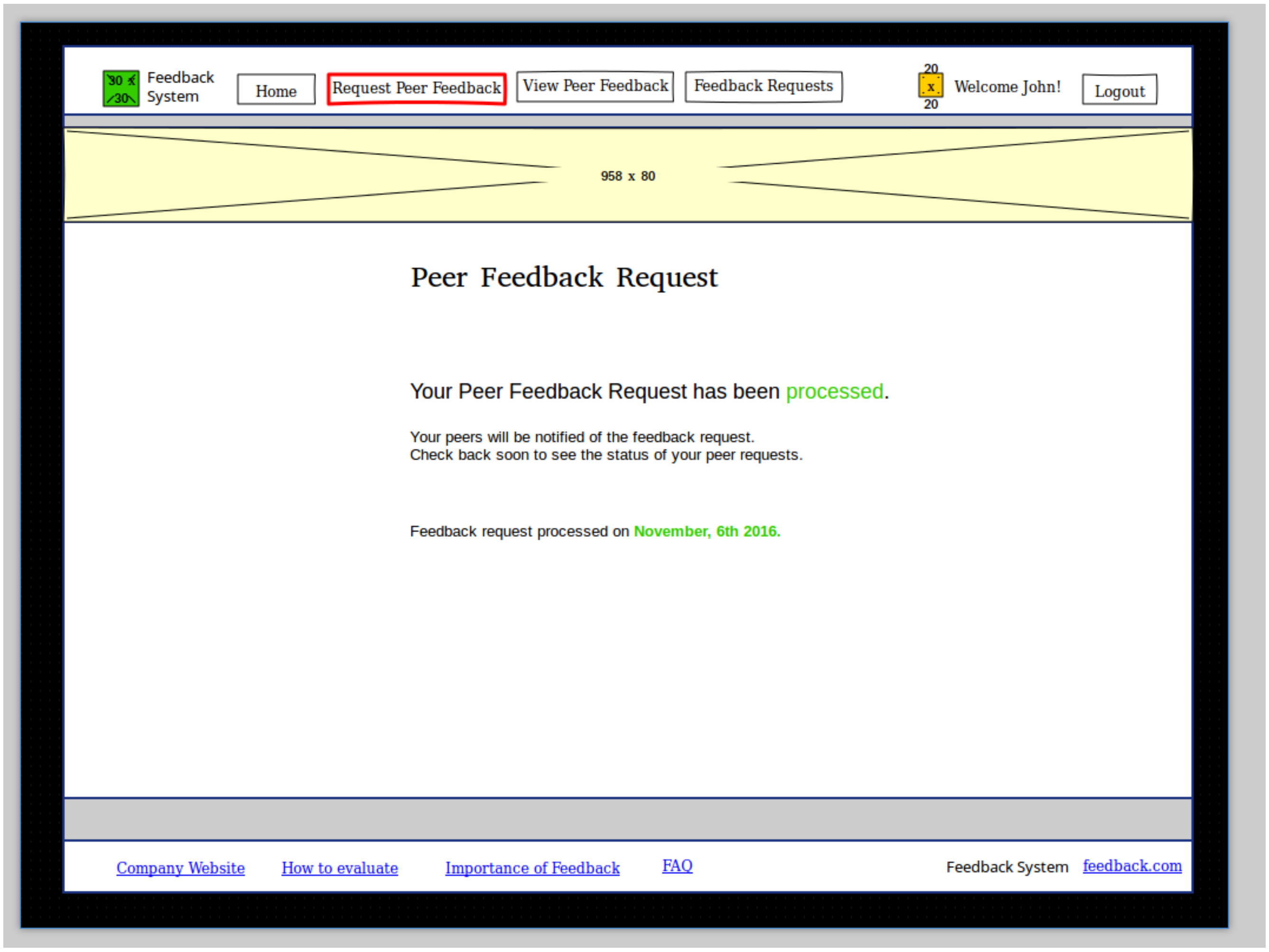
....

....

....

Do you accept these
Terms and Conditions

☐ I accept

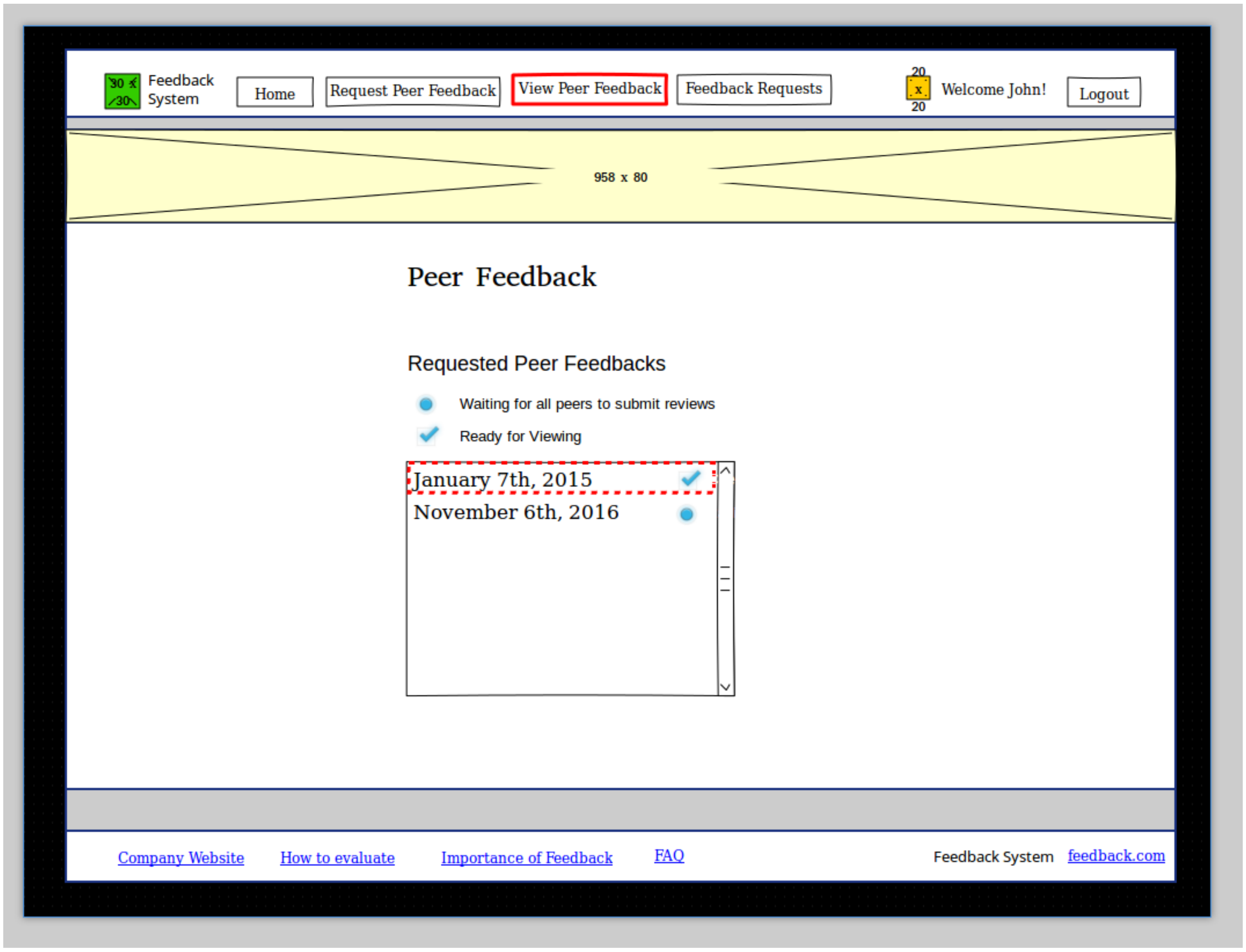


Peer Feedback Request

Your Peer Feedback Request has been processed.

Your peers will be notified of the feedback request.
Check back soon to see the status of your peer requests.

Feedback request processed on **November, 6th 2016.**



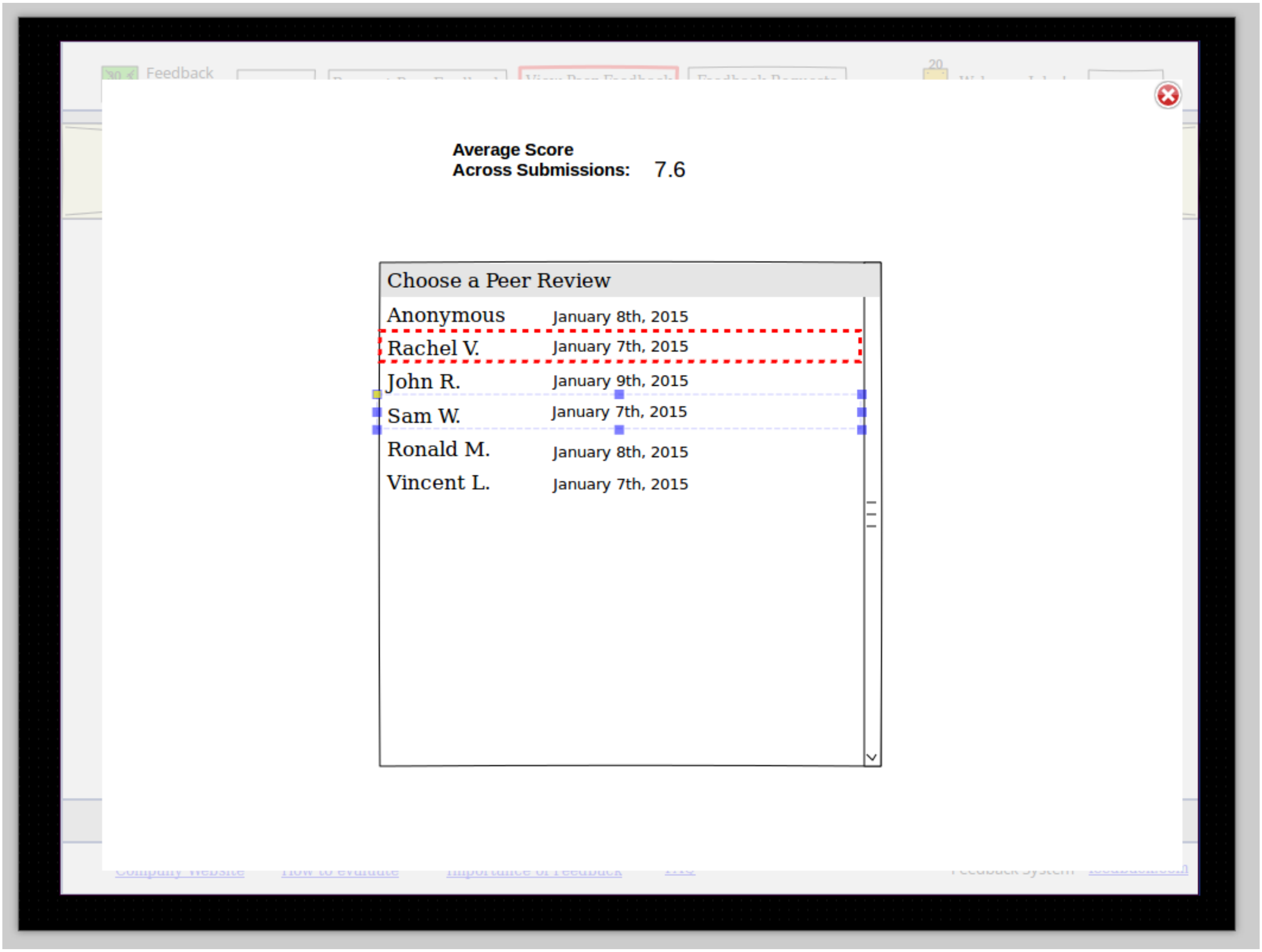
Peer Feedback

Requested Peer Feedbacks

● Waiting for all peers to submit reviews

✓ Ready for Viewing

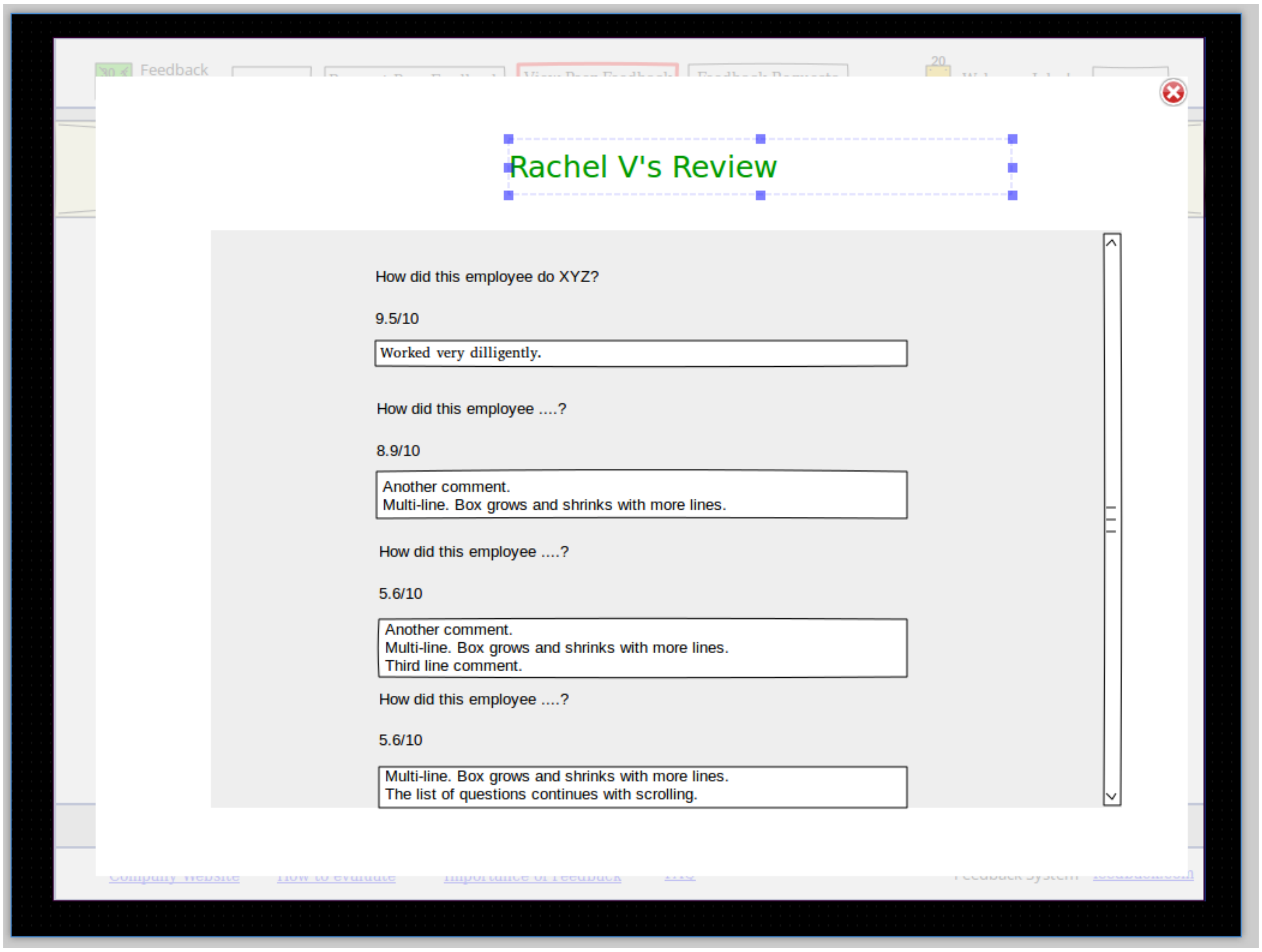
January 7th, 2015	✓
November 6th, 2016	●



Average Score
Across Submissions: 7.6

Choose a Peer Review

Anonymous	January 8th, 2015
Rachel V.	January 7th, 2015
John R.	January 9th, 2015
Sam W.	January 7th, 2015
Ronald M.	January 8th, 2015
Vincent L.	January 7th, 2015



Rachel V's Review

How did this employee do XYZ?

9.5/10

Worked very dilligently.

How did this employee ....?

8.9/10

Another comment.
Multi-line. Box grows and shrinks with more lines.

How did this employee ....?

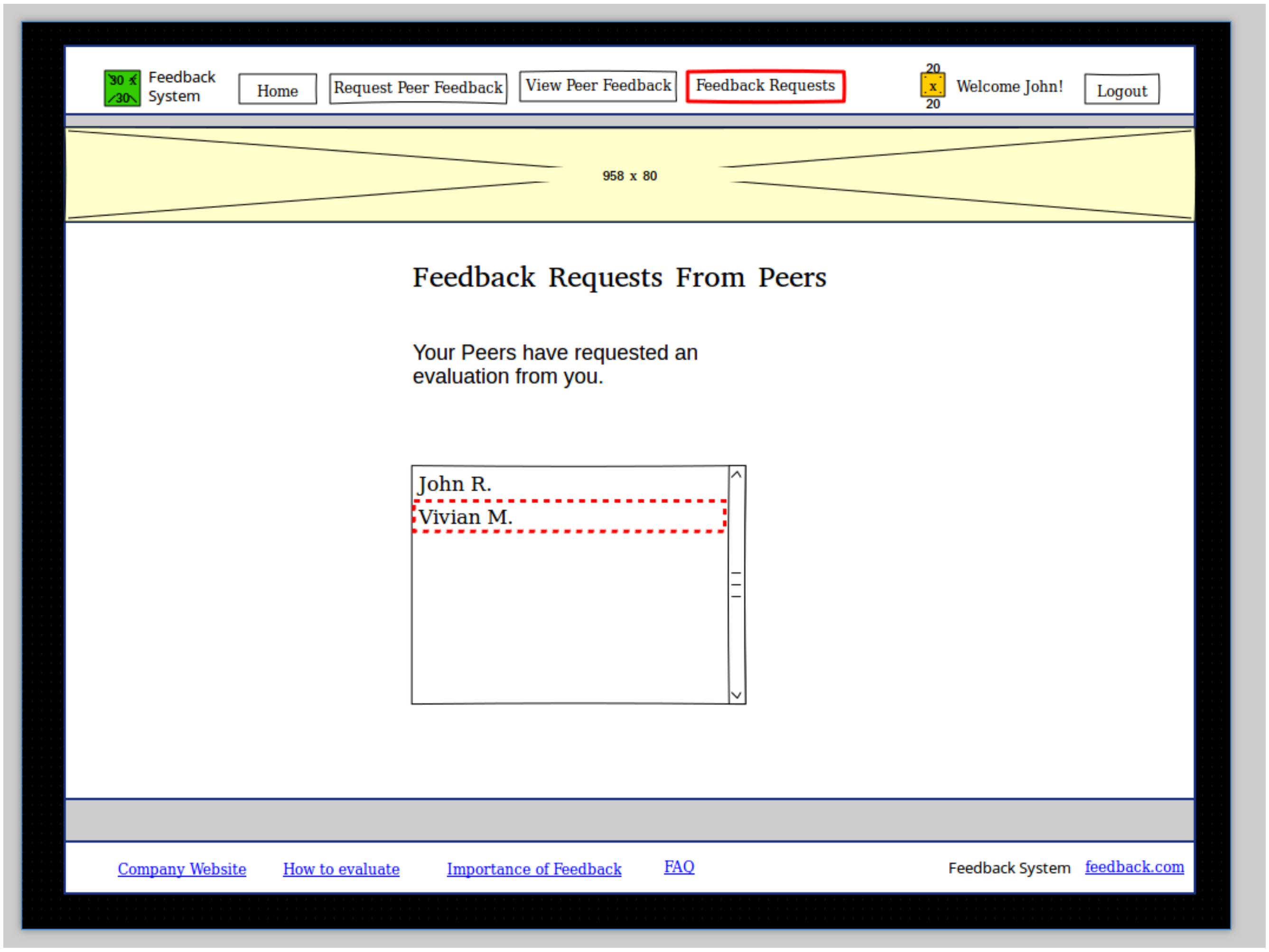
5.6/10

Another comment.
Multi-line. Box grows and shrinks with more lines.
Third line comment.

How did this employee ....?

5.6/10

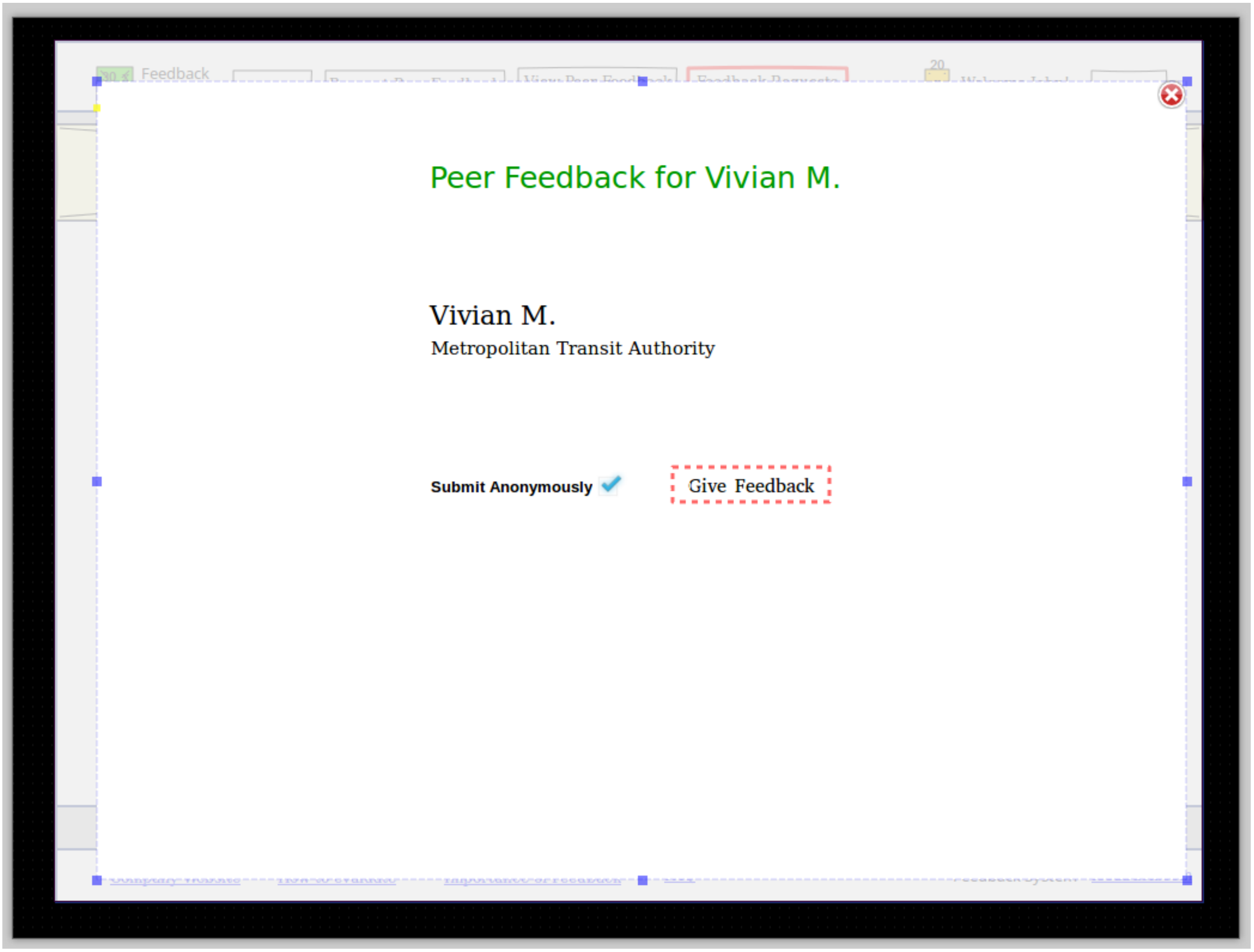
Multi-line. Box grows and shrinks with more lines.
The list of questions continues with scrolling.



Feedback Requests From Peers

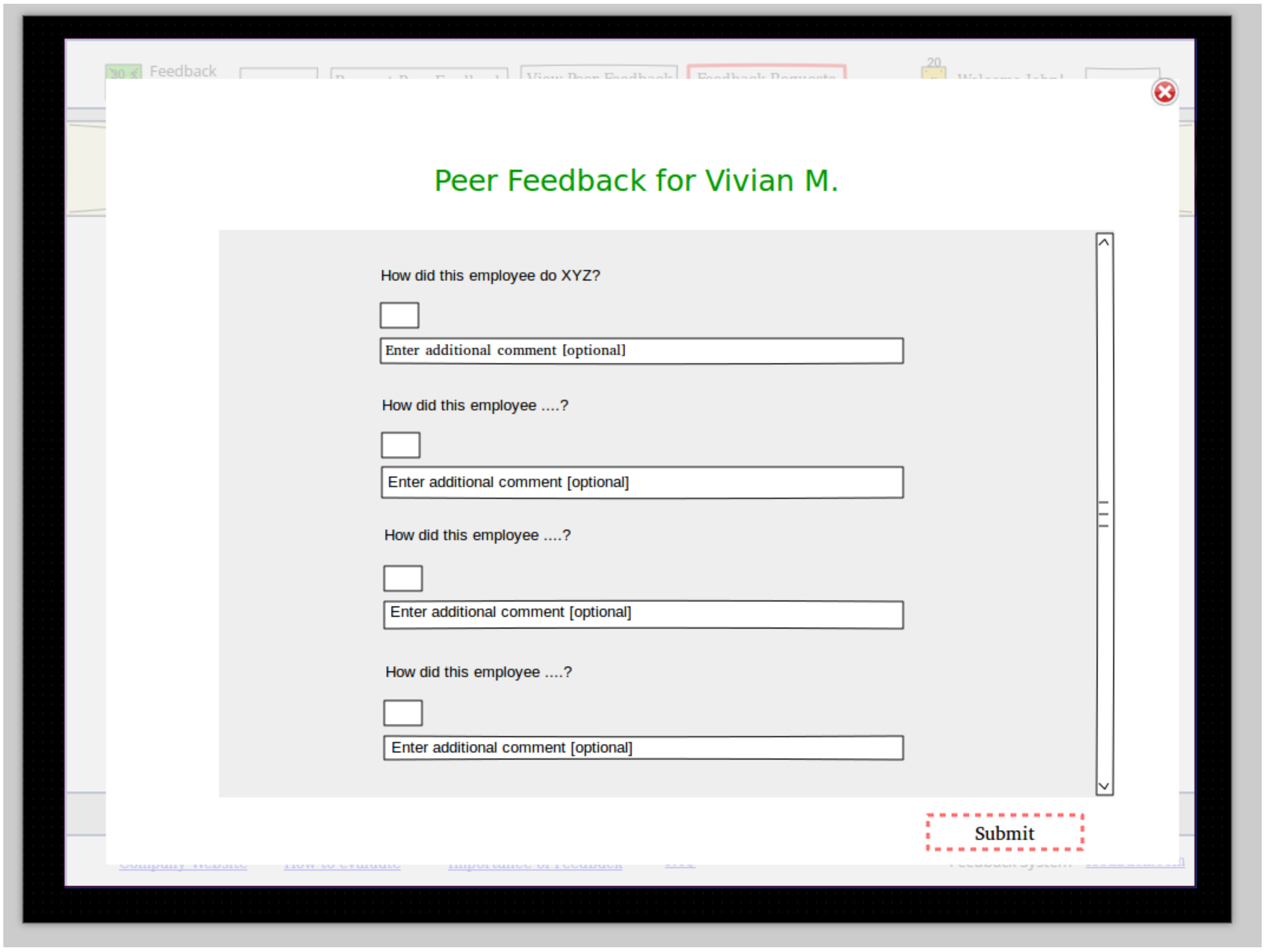
Your Peers have requested an evaluation from you.

John R.
Vivian M.



Peer Feedback for Vivian M.

Vivian M.
Metropolitan Transit Authority



Peer Feedback for Vivian M.

How did this employee do XYZ?

Enter additional comment [optional]

How did this employee ....?

Enter additional comment [optional]

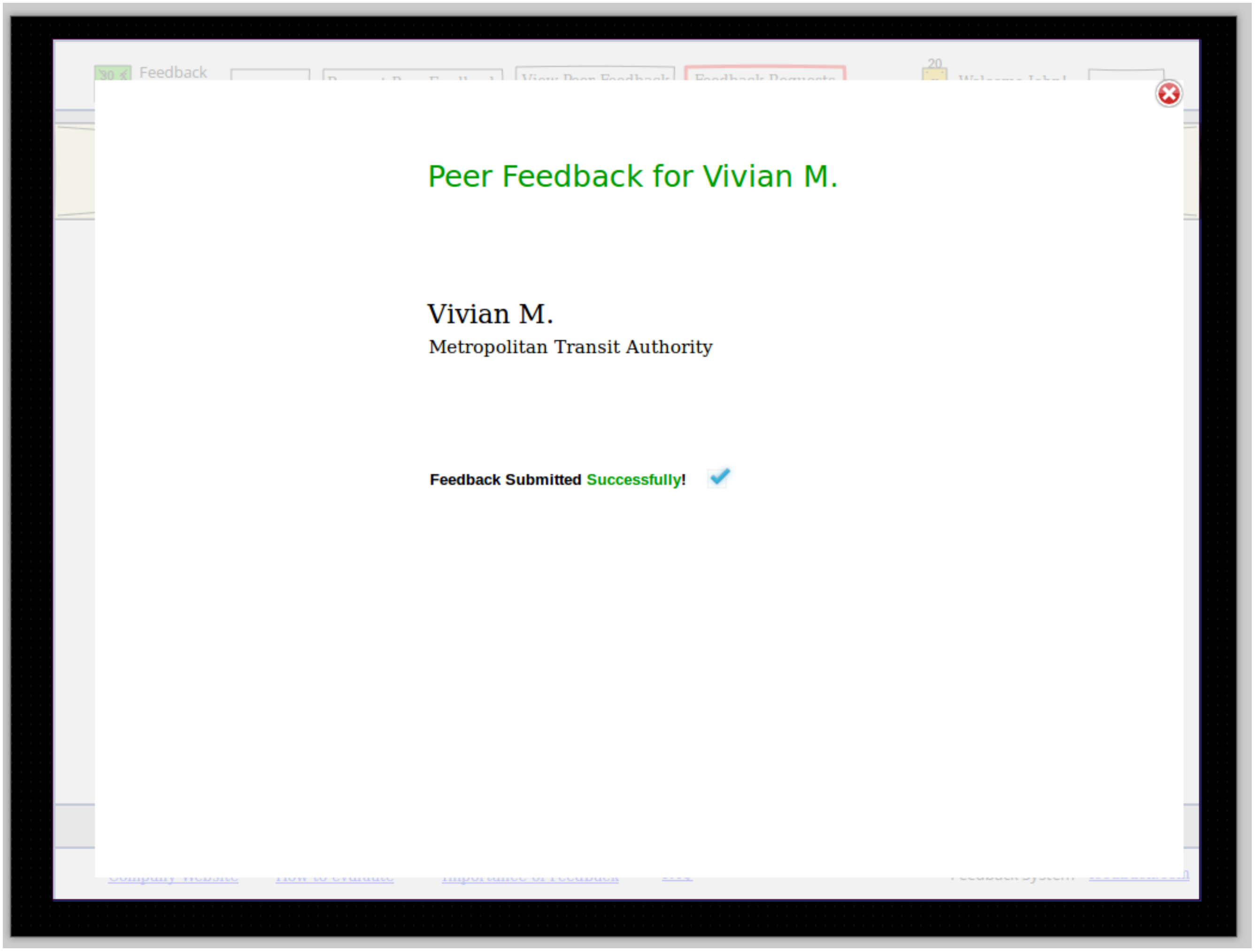
How did this employee ....?

Enter additional comment [optional]

How did this employee ....?

Enter additional comment [optional]

Submit

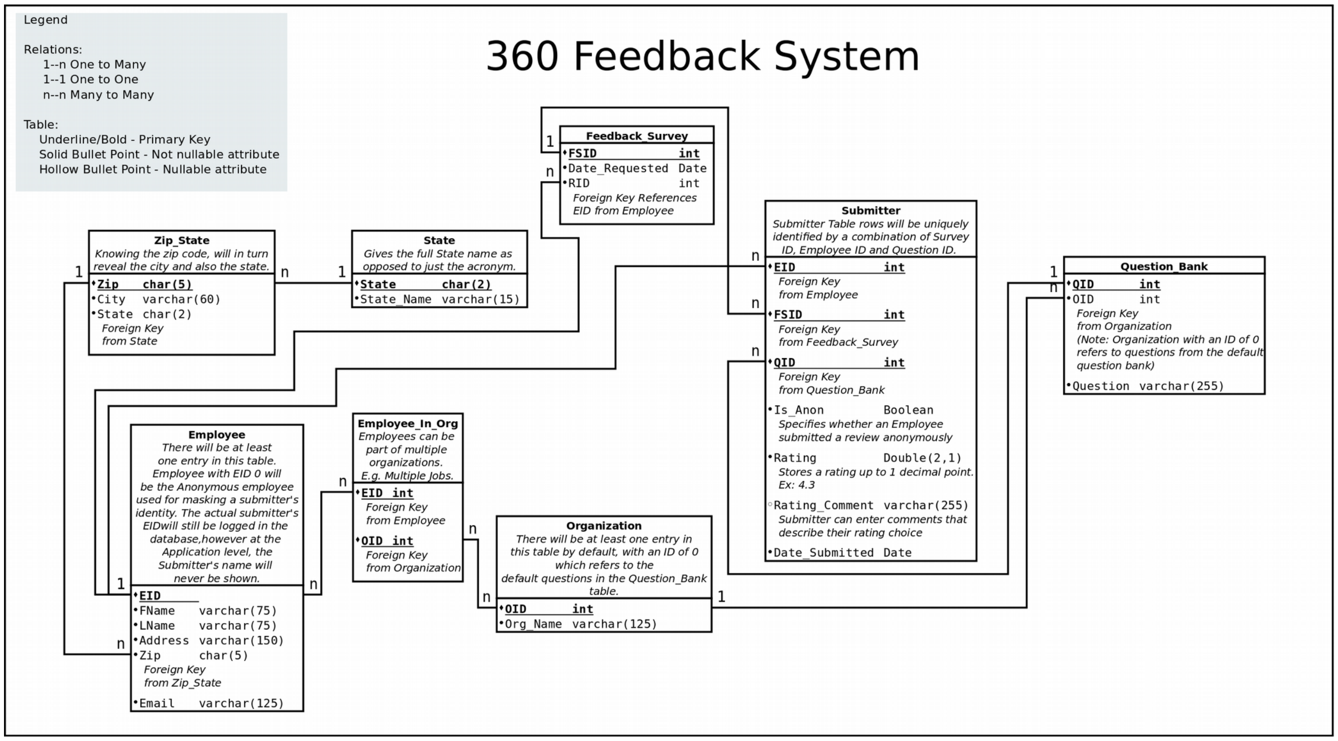


Peer Feedback for Vivian M.

Vivian M.
Metropolitan Transit Authority

Feedback Submitted Successfully! ✓

Complete Physical ERD


Legend
Relations:
1--n One to Many
1--1 One to One
n--n Many to Many
Table:
Underline/Bold - Primary Key
Solid Bullet Point - Not nullable attribute
Hollow Bullet Point - Nullable attribute
360 Feedback System
Feedback_Survey
FSID int
Date_Requested Date
RID int
Foreign Key References EID from Employee
Submitter
Submitter Table rows will be uniquely identified by a combination of Survey ID, Employee ID and Question ID.
EID int
Foreign Key from Employee
FSID int
Foreign Key from Feedback_Survey
QID int
Foreign Key from Question_Bank
Is_Anon Boolean
Specifies whether an Employee submitted a review anonymously
Rating Double(2,1)
Stores a rating up to 1 decimal point. Ex: 4.3
Rating_Comment varchar(255)
Submitter can enter comments that describe their rating choice
Date_Submitted Date
Question_Bank
QID int
OID int
Foreign Key from Organization
(Note: Organization with an ID of 0 refers to questions from the default question bank)
Question varchar(255)
Zip_State
Knowing the zip code, will in turn reveal the city and also the state.
Zip char(5)
City varchar(60)
State char(2)
Foreign Key from State
State
Gives the full State name as opposed to just the acronym.
State char(2)
State_Name varchar(15)
Employee
There will be at least one entry in this table. Employee with EID 0 will be the Anonymous employee used for masking a submitter's identity. The actual submitter's EIDwill still be logged in the database,however at the Application level, the Submitter's name will never be shown.
EID
FName varchar(75)
LName varchar(75)
Address varchar(150)
Zip char(5)
Foreign Key from Zip_State
Email varchar(125)
Employee_In_Org
Employees can be part of multiple organizations. E.g. Multiple Jobs.
EID int
Foreign Key from Employee
OID int
Foreign Key from Organization
Organization
There will be at least one entry in this table by default, with an ID of 0 which refers to the default questions in the Question_Bank table.
OID int
Org_Name varchar(125)
1
n

Note:

* Assumptions given in the ER diagram.
* Note that the company gets to choose what questions are given in the feedback survey. This is handled by an administrative entity, and was not included in the wire frame since the assumption is that this would be done at the setup of the application.

Database Query

Retrieves Question Text, Submitter’s Response and the Name of the Submitter only if it’s not Anonymous. If so, display ‘Anonymous’ for the Name.

```
Select a.Question,
        CONCAT(b.Rating, ' | ', b.Rating_Comment) as Submitter_Response,
        case b.Is_Anon
                when 1 then CONCAT(c.Fname,', ', c.Lname)
                when 0 then 'Anonymous' end as Submitter_Name
        FROM
        Feedback_Survey d INNER JOIN Submitter b
                on d.FSID = b.FSID INNER JOIN Question_Bank a
                on b.QID = a.QID INNER JOIN Employee c
                on c.EID = b.EID
        WHERE d.FSID = 2;
```

Algorithm

```
Hashmap<String, double> getAverageScores(userID, questionsPerSurvey){

    create Hashmap<String, Hashmap<String, double> dataQueryHolder;

    select rating entries from database
    where the userID is found in the Feedback_Survey table
    and where the submitters submitted to a surveyID
        that is associated with the userID.
    Retrieve all the ratings per submitter, per surveyID
    that were associated with userID

    create double ratingSum=0;
    create ratingCounter=0;
    create double mean=0;

    Store into dataQueryHolder
    where the surveyID will by the key to the outer hash map
        and where submitterID will by the key to the inner hash map
        with a value of Rating.
    While storing each rating{
        ratingSum += rating;
        ratingCounter++;
    }
    once finished,
    mean = (ratingSum/ratingCounter);

    Create double standardDeviation=0;

    create Array<double> tempForStandDev[ratingCounter];
    create int counterForSD;

    //calculate the standardDeviation
    for each(surveyID in dataQueryHolder){
        for each(rating in submitterID){
            tempForStandDev[counterForSD]= squared(rating - mean);
            counterForSD++;
        }
    }

    create double sumForStandDev;
    for each(entry in tempForStandDev){
        sumForStandDev += entry;
    }

    //Standard Deviation for ALL the ratings from the database query for userID
```

```
        standardDeviation = sqrt(sumForStandDev/ratingCounter);



        //Normalizing each score
        create double avgOfSubmitters;
        create Hashmap<String, double> avgPerSurvey;
        for each(surveyID in dataQueryHolder){
                create double sumPerSurveyTemp;
                for each(rating in submitterID){
                        rating = (rating – mean)/(standardDeviation);
                        avgOfSubmitters += rating;
                        sumPerSurveytemp += rating;
                }
                sumPerSurveyTemp /= questionsPerSurvey;
                String hashKey = surveyID + “avg”;
                avgPerSurvey.insert(hashkey, sumPerSurveyTemp);
        }
        avgOfSubmitters /= ratingCounter;

        create Hashmap<String, double> returnsAveragedData;


        returnsAveragedData.insert(“avgAcrossSubmitters”, avgOfSubmitters);
        for each(entry in avgPerSurvey){
                returnsAveragedData .insert(each entry with same key, value);
        }
        for each(surveyID in dataQueryHolder){
                int mixinHashValue = 0;
                for each(rating in submitterID){
                        String newHashKey = surveyID + submitterID + mixinHashValue;
                        returnsAveragedData.insert(newHashKey, rating);
                }
        }

        /*
          Returns all  (normalized) submitted feedback, along with
          average score across all submitters and all of the feedback.
        */

        return returnsAveragedData;


}
```