RateMyClassmate

By MergeMonkeys
(Nabil Amiri, Arash Zahoory,
Cody Klein, & Brandon Rossi)

Functional Specifications for
Rate My Classmate Application

Group: Merge Monkeys

Problem Statement

The purpose of this project is to develop a web application to rate group members and their performance during group projects.

Objectives

This application should be able to interact with a user and allow them to rate classmates based upon their perceived performance within projects. Additionally, users should be able to access the website and view previously submitted reviews.

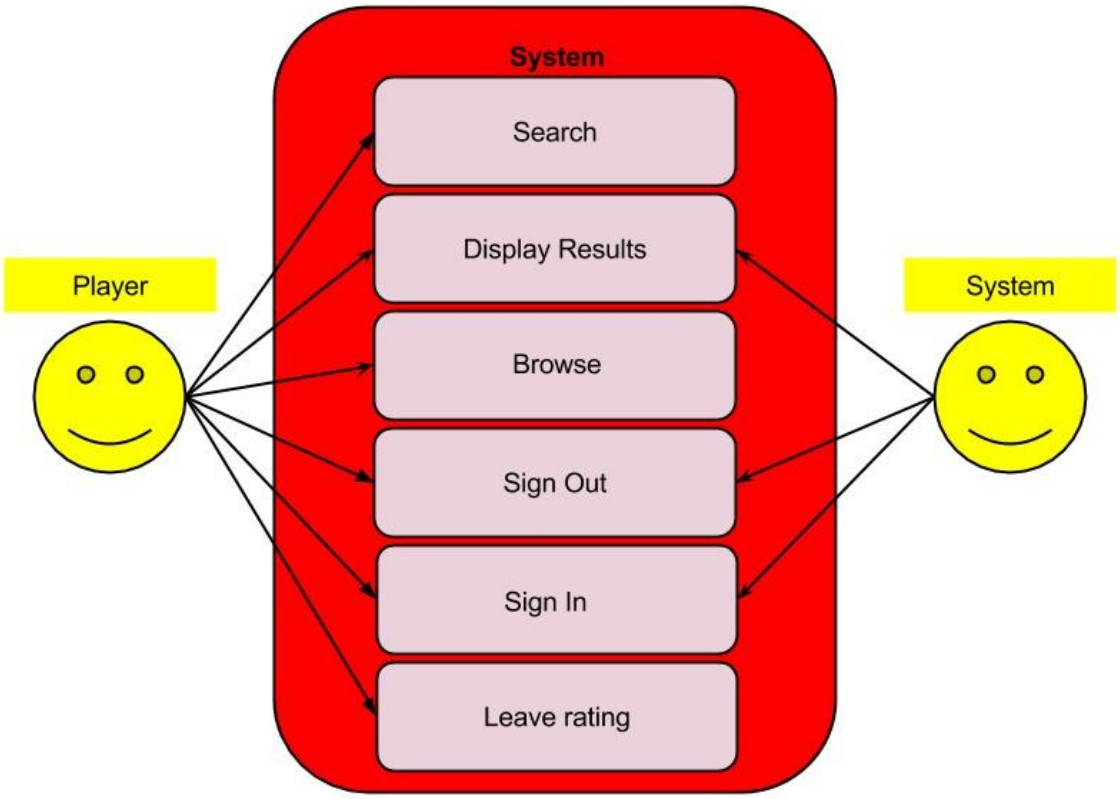
Functional Requirements

REQ-1: SYSTEM shall prompt the USER for a students name
REQ-2: USER may browse complete list of user rating profiles
REQ-3: SYSTEM shall display results to users based on database queries.
REQ-4: USER can sort results based on different categories.
REQ-5: USER has ability to validate other reviews with upvote/downvote system.
REQ-6: USER has ability to confront own reviews.
REQ-7: USER accessing the web application will not be able to edit previously submitted reviews

Non-functional Requirements

NFR-1: SYSTEM should be accessible from any device (Android, iOS, Mac OS, Windows)
NRF-2: USER profile password must be no less than seven (7) characters long
NRF-3: Reviews that are posted will not allow for malicious or degrading commentary. Furthermore, no personal information will be allowed. All content will be filtered.
NRF-4: Only plaintext commentary will be allowed onto the website. No images or other such content will be allowed to be submitted.

Use Case Diagram



Use Case Description #1

Use Case name:	Search reviews
Project name:	Rate My Classmate
Team:	Merge Monkeys
Date:	15 September 2014

1. Goal

· Find reviews based on the name of a classmate

2. Summary

· User enters name and result will be displayed

3. Actors

· User
· System

4. Preconditions

· Search field must have valid text

5. Trigger

· User clicks the search button

6. Primary Sequence	
Step	**Action**
1	User searches name
2	System searches database for name
3	System provides results matching name

7. Primary Postconditions

· User ability to select desired classmate

8. Alternate Sequences	User enters invalid name
Step	**Action**
1.1	User searches name
1.2	System cannot find name in database
1.3	System returns error to User: Prompts user to try check spelling or create new profile for potential recipient of intended

	review
Alternate Trigger	User enters invalid data in search field. e.g. User enters numbers, non alpha characters, or nonexistent profile name.

Use Case Description #2

Use Case name:	Leave a review
Project name:	Rate My Classmate
Team:	Merge Monkeys
Date:	15 September 2014

1. Goal

· To successfully submit a review

2. Summary

· The review that the user submits will be added to the web app's database of reviews

3. Actors

· User
· System

4. Preconditions

· A user profile must exist for the recipient of the review

5. Trigger

· Once the review has been entered the user clicks the submit button

6. Primary Sequence	
Step	**Action**
1	User selects recipient of review
2	User writes review in the provided space
3	User clicks submit review button
4	System acknowledges successful submission of review
5	System displays review alongside recipients other review

7. Primary Postconditions

· Main menu is displayed on the screen

8. Alternate Sequences	User does not enter any text in the required fields and clicks submit
Step	**Action**
2.1	User clicks submit before entering a review
2.2	System returns an error informing user that they have to fill out required fields
2.3	User restarts use case number two

Use Case Description #3

Use Case name:	Upvote a review
Project name:	Rate My Classmate
Team:	Merge Monkeys
Date:	15 September 2014

1. Goal

· Allow users to validate a review

2. Summary

· User will be able to endorse a review based upon the accuracy of the review

3. Actors

· User
· System

4. Preconditions

· There must be a review regarding a classmate
· User must be viewing reviews for a specific classmate

5. Trigger

· User clicks the upvote button associated with a review

6. Primary Sequence	
Step	**Action**
1	User clicks the up arrow icon above the review
2	System provides message confirming upvote

3	System increments vote counter

7. Primary Postconditions

· The same user can not vote again on this specific review

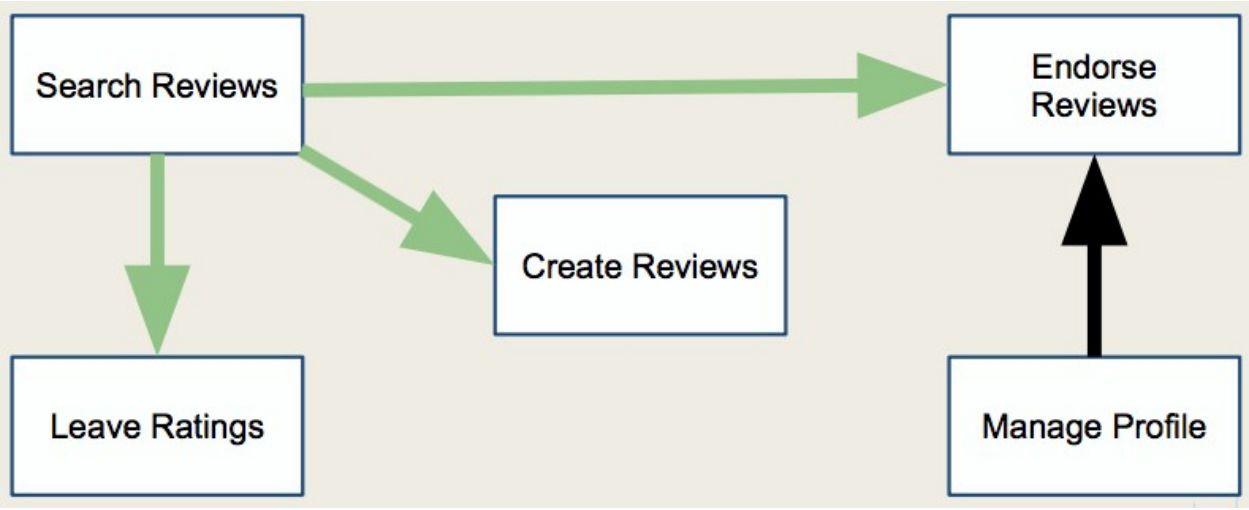
8. Alternate Sequences	None
Alternate Trigger	None

Conceptual Design

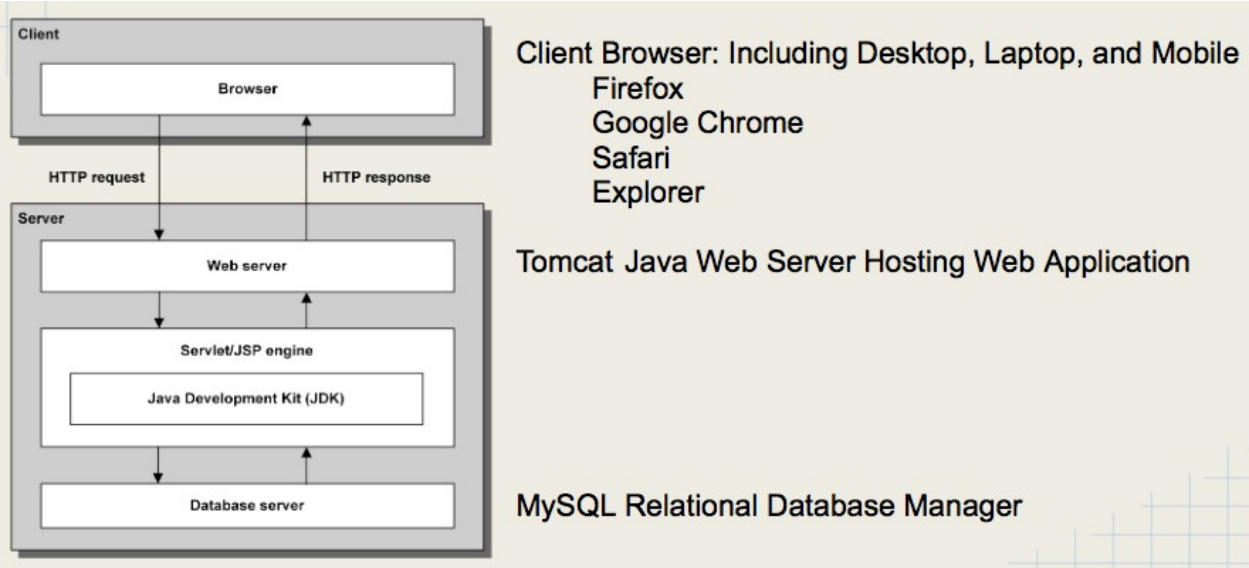
Major Features:

- User Interface
  - Application interface will be simple to allow for access from multiple platforms --- Desktop, Mobile, Tablets, etc.
- Create Reviews
  - Students will be able to create placeholders for other students, in order to record ratings and leave reviews for people who lack profiles.
  - Written reviews will be a way for students to recognize actual experiences that they shared with their classmates.
  - Rate fellow classmates group behavior based on Availability, Intelligence, Motivation, Dependability, Friendliness, Creativity.
- Search for Students and Classmates
  - The ability to search for classmates allows students to observe the reviews already present for that person.
  - Reviews for people are private unless user that is viewing also shares access to their own reviews and review history.
- Leave Ratings
  - Ratings are a quick way to voice recognition of a group members behavior during the semester, using just a few common parameters.
- Endorse Ratings
  - Other students can endorse the rating of their classmates by agreeing or disagreeing with what is written by their classmates.

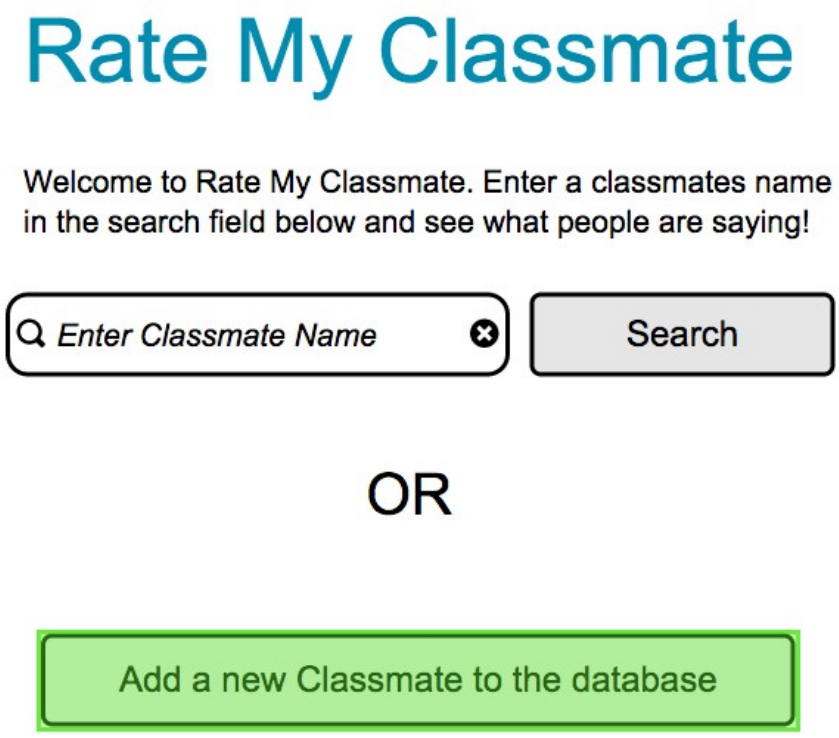
Major Modules:



High-level Architecture Diagrams:

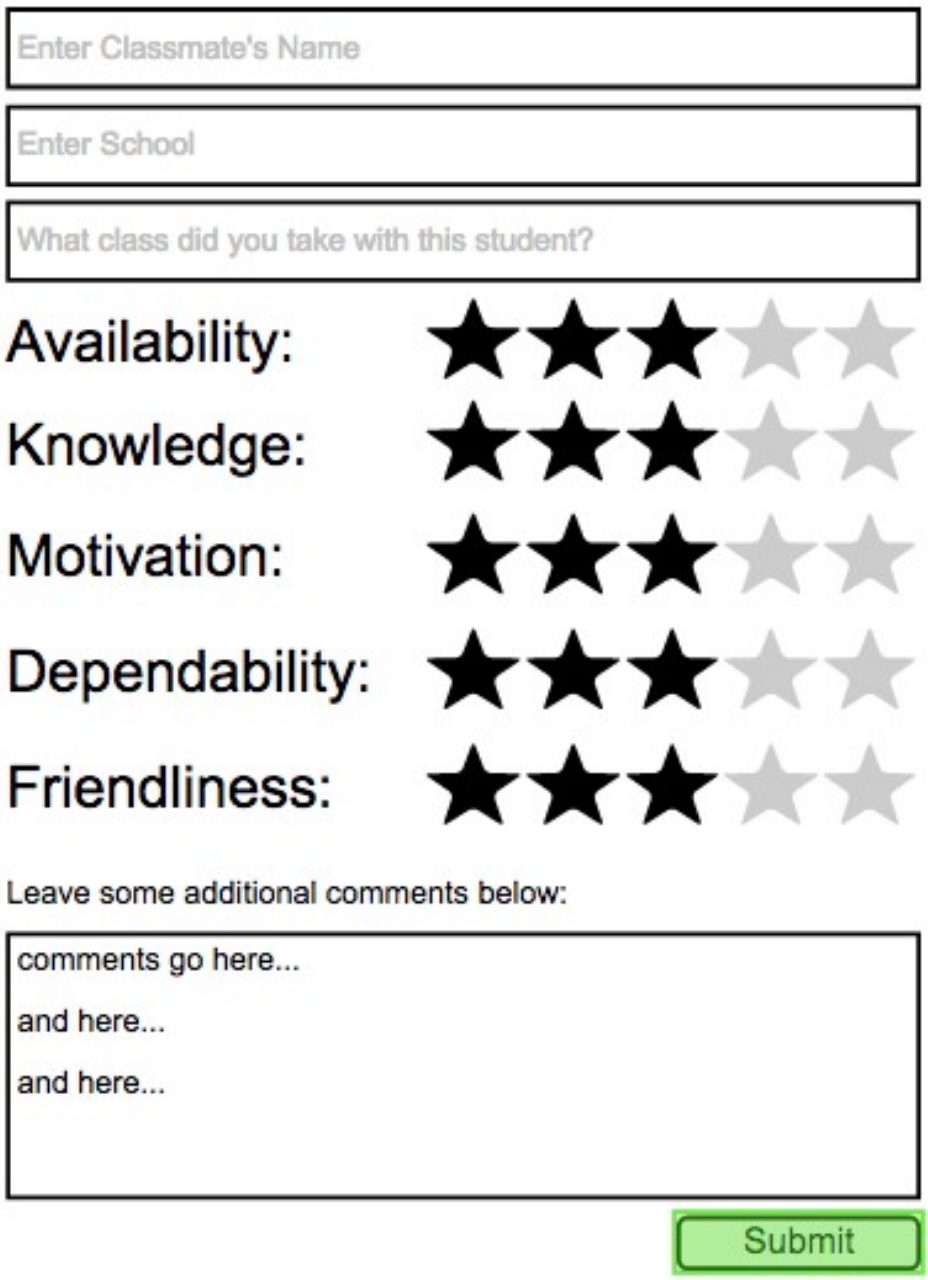


Webpage Mock-Up:



Add a new classmate

This feature is meant to be used if a user was not found in the database.

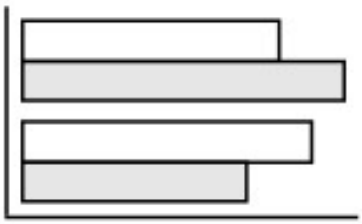


John Doe

Overall Rating: ★★★☆☆

12px

Friendliness:
Availability:
Knowledge:
Dependability:
Dependability:



Reviewer 1

Multi
line
textarea

Reviewer 2

Multi
line
textarea

Reviewer 3

Multi
line
textarea

Reviewer 4

Multi
line
textarea

UpdateProfile - Controller
This class contains two primary variables; name: String, school: School. The **name** variable holds the String value of the name of the profile brought in by the constructor, and the **school** object holds the details of the school that the profile of the user belongs to. The methods UpdateProfile, and createProfile are used to create and update the data involved in what distinguishes user profiles. UpdateProfile is a constructor. The primary objective of this class is to update the database using ModelProfile objects.

SearchUser - Controller
This class contains one primary variable; user: User. This variable holds the details of the user given to set by the constructor SearchUser. The other method **search** is used to query the database for the search results, and to notify the display to update with these results. The primary objective of this class is to update the search results that are displayed by the DisplaySearchResults objects.

UpdateReviews - Controller
This class contains one primary variable; review: Review. This Review object retains the details of the review that is set by the constructor and will be sent to the database. The three methods currently available for this class are UpdateReviews, deleteReview, and createReview. UpdateReviews is the constructor. The objective of this class is to use its methods to update the database regarding review data.

UpdateRating - Controller
This class contains a single variable; rating: Rating. This Rating object retains the details regarding the rating that is set by the constructor, UpdateRating. The methods currently available for this class are UpdateRating, UpdateUserRating, and checkRatingExistence. The purpose of this class is to update the database regarding the ratings of users using these methods.

ModelProfile - Model
This class contains one variable; user: User. This User object retains the details regarding the user that is set by ModelProfile, which is the constructor. The methods provided in this class include ModelProfile, updateProfile, and updateView. The purpose of this class is to keep track of the profile data, and to notify the DisplaySearchResults, DisplayUserReviews, and DisplayProfile objects to accurately portray current profile data to the user.

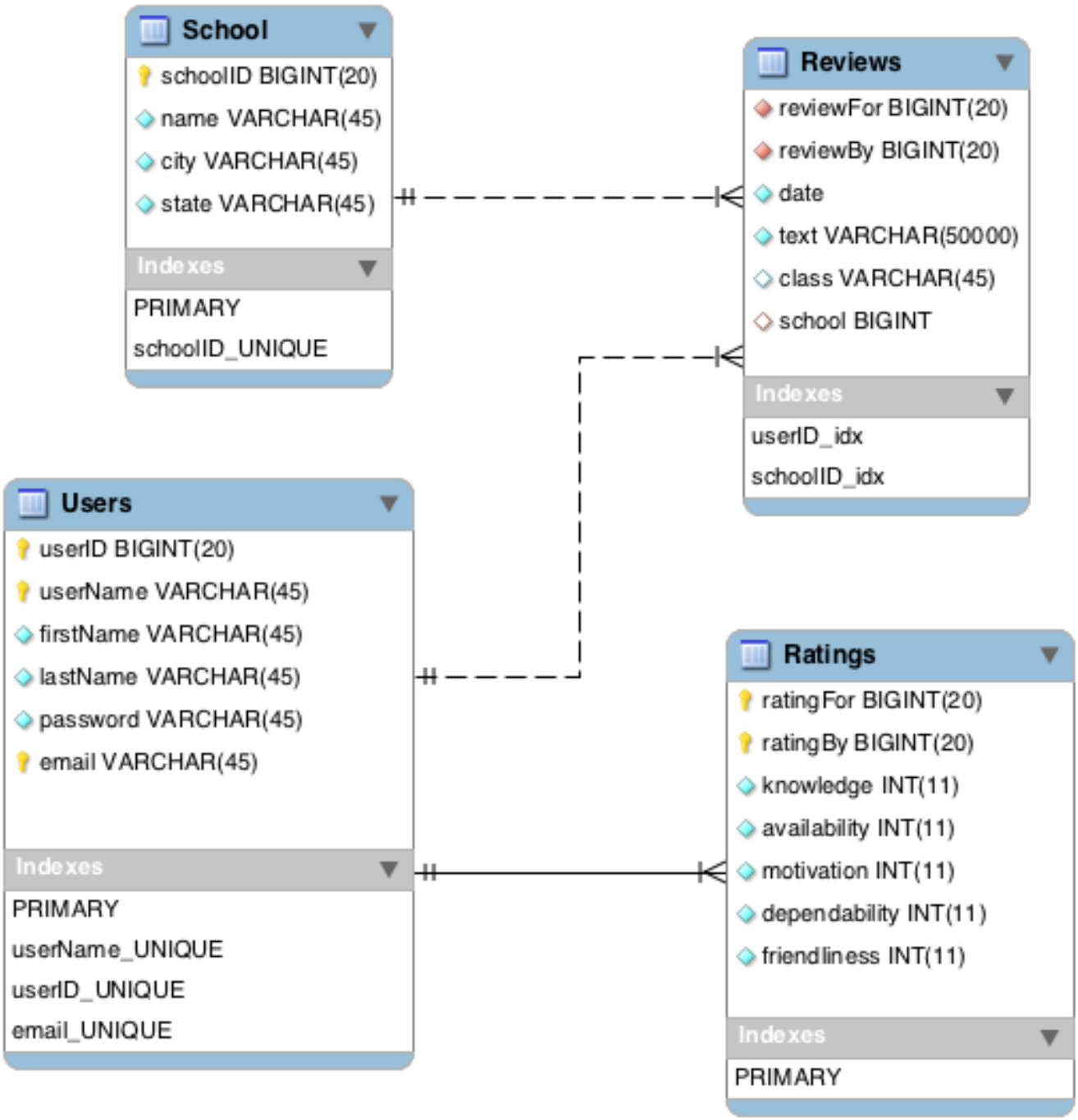
ModelReview - Model
This class contains one variable; user: User. This User object retains the details regarding the user that is set by ModelReview, which is the constructor. The methods provided in this class include ModelReview, updateReview, and updateView. The purpose of this class is to keep track of the review data, and to notify the DisplayUserReviews, and DisplayProfile objects to accurately portray current review data to the user.

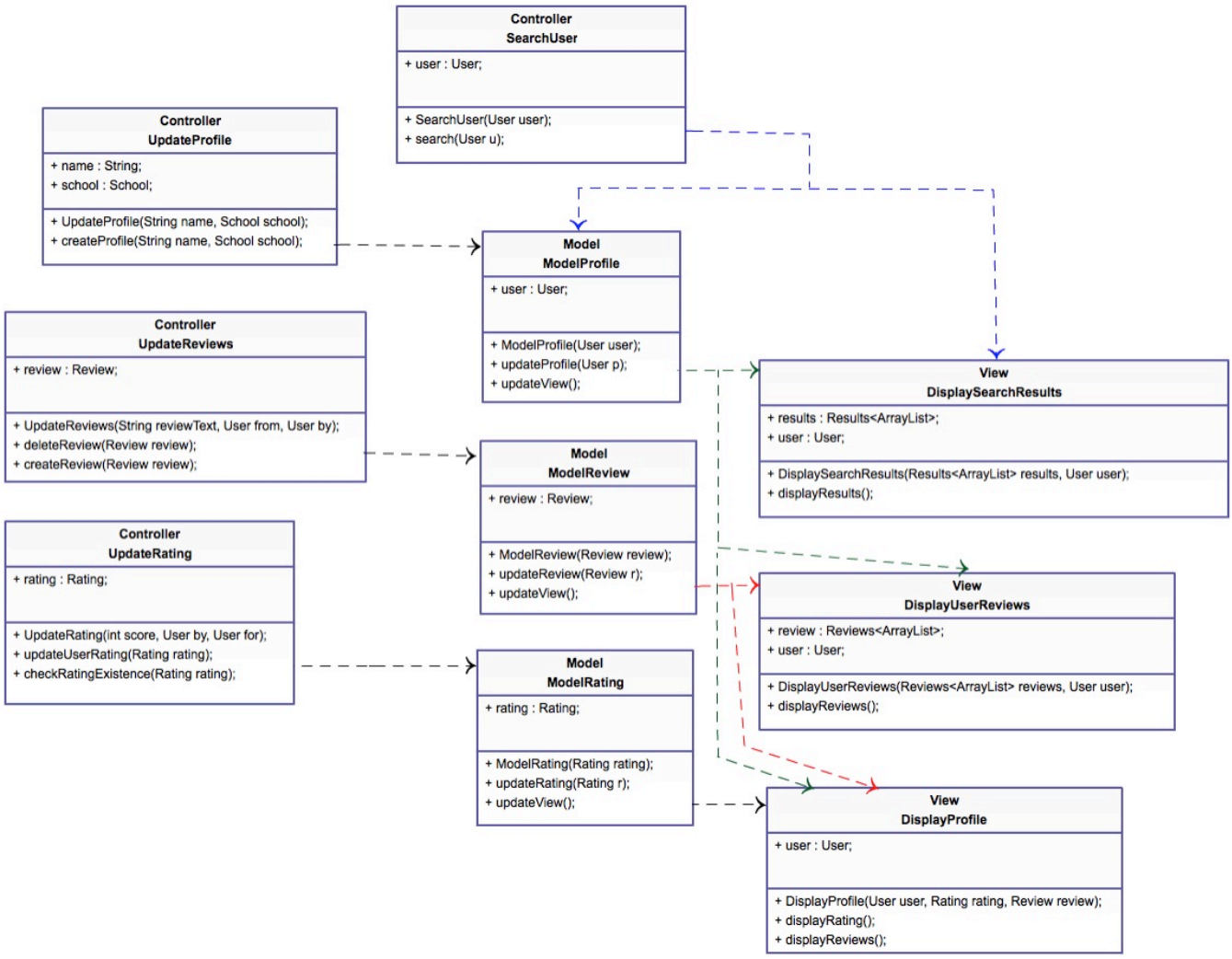
ModelRating - Model
This class contains one variable; user: User. This User object retains the details regarding the user that is set by ModelRating, which is the constructor. The methods provided in this class include ModelRating, updateProfile, and updateView. The purpose of this class is to keep track of the rating data, and to notify the DisplayProfile object to accurately portray current rating data to the user.

DisplaySearchResults - View
This class contains two variables; results: Results<ArrayList>, and user: User. The Results array list is a list of result objects that an object of this class will use to display results to the user. The methods in this class are DisplaySearchResults and displayResults. DisplaySearchResults is a constructor that sets the results array list, and the user variable (which details the user which will be displayed). The purpose of this class is to display to the user the search results that are provided to it by the database.

DisplayProfile - View
This class contains one variable; user: User. The user object holds the details of the user for which the profile will be displayed, and it is set by the constructor DisplayProfile. The methods in this class include DisplayProfile, displayRating, and displayReviews. The purpose of this class is to display a user profile to the user using the data provided to it by the database.

DisplayReviews - View
This class contains has two variables; user: User, and review: Reviews<ArrayList>. The Reviews array list is a list of Review objects that an object of this class will use to display user reviews to the user. The purpose of this class is to display user reviews to the user using the data provided to it by the database.


School
schoolID BIGINT(20)
name VARCHAR(45)
city VARCHAR(45)
state VARCHAR(45)
Indexes
PRIMARY
schoolID_UNIQUE
Reviews
reviewFor BIGINT(20)
reviewBy BIGINT(20)
date
text VARCHAR(50000)
class VARCHAR(45)
school BIGINT
Indexes
userID_idx
schoolID_idx
Users
userID BIGINT(20)
userName VARCHAR(45)
firstName VARCHAR(45)
lastName VARCHAR(45)
password VARCHAR(45)
email VARCHAR(45)
Indexes
PRIMARY
userName_UNIQUE
userID_UNIQUE
email_UNIQUE
Ratings
ratingFor BIGINT(20)
ratingBy BIGINT(20)
knowledge INT(11)
availability INT(11)
motivation INT(11)
dependability INT(11)
friendliness INT(11)
Indexes
PRIMARY


Controller
SearchUser
+ user : User;
+ SearchUser(User user);
+ search(User u);
Controller
UpdateProfile
+ name : String;
+ school : School;
+ UpdateProfile(String name, School school);
+ createProfile(String name, School school);
Model
ModelProfile
+ user : User;
+ ModelProfile(User user);
+ updateProfile(User p);
+ updateView();
Controller
UpdateReviews
+ review : Review;
+ UpdateReviews(String reviewText, User from, User by);
+ deleteReview(Review review);
+ createReview(Review review);
Model
ModelReview
+ review : Review;
+ ModelReview(Review review);
+ updateReview(Review r);
+ updateView();
Controller
UpdateRating
+ rating : Rating;
+ UpdateRating(int score, User by, User for);
+ updateUserRating(Rating rating);
+ checkRatingExistence(Rating rating);
Model
ModelRating
+ rating : Rating;
+ ModelRating(Rating rating);
+ updateRating(Rating r);
+ updateView();
View
DisplaySearchResults
+ results : Results<ArrayList>;
+ user : User;
+ DisplaySearchResults(Results<ArrayList> results, User user);
+ displayResults();
View
DisplayUserReviews
+ review : Reviews<ArrayList>;
+ user : User;
+ DisplayUserReviews(Reviews<ArrayList> reviews, User user);
+ displayReviews();
View
DisplayProfile
+ user : User;
+ DisplayProfile(User user, Rating rating, Review review);
+ displayRating();
+ displayReviews();

Merge Monkeys ‘Review’ Sequence Diagram

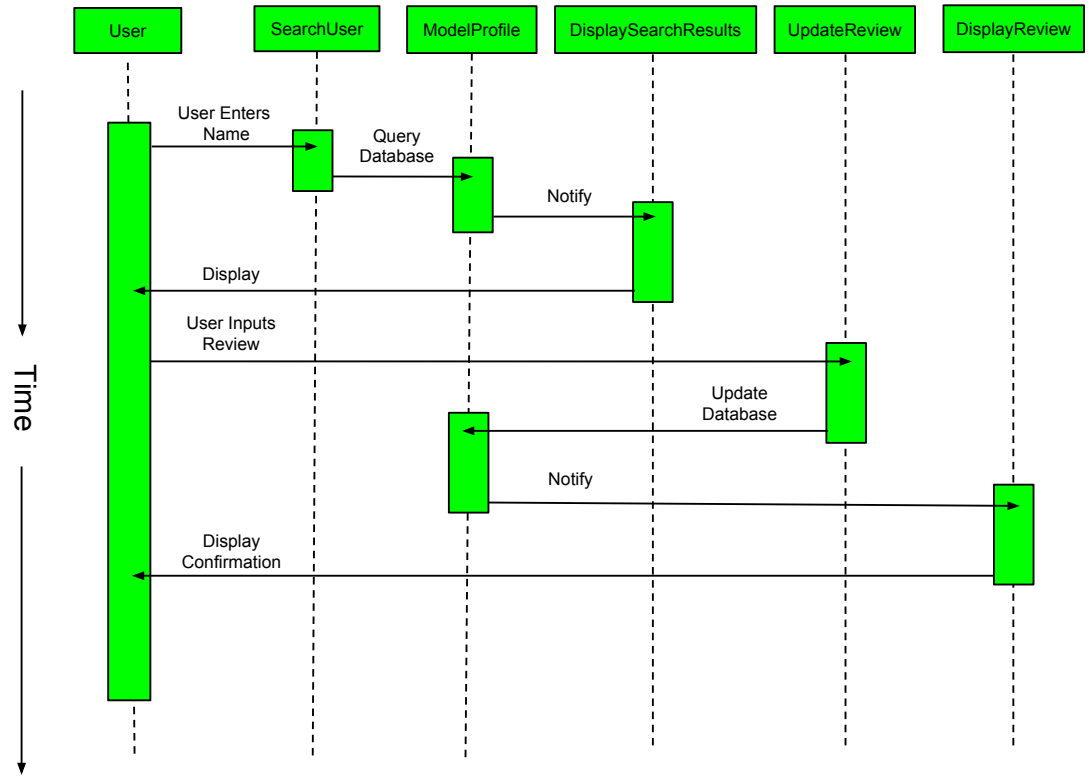
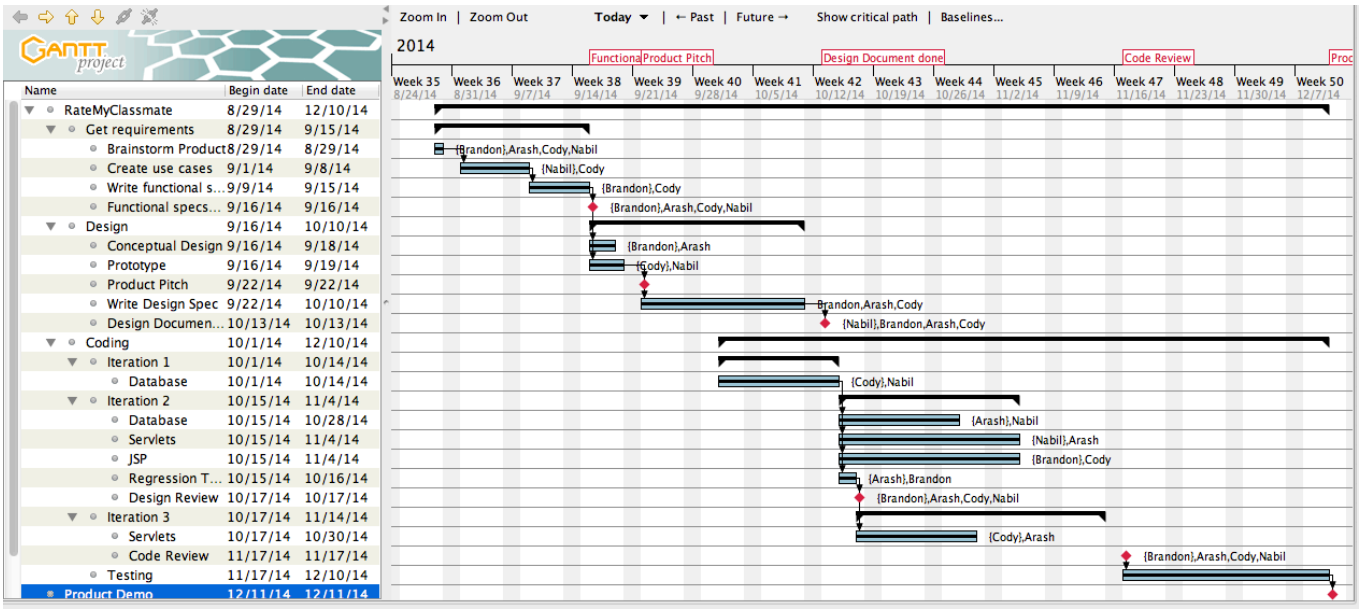
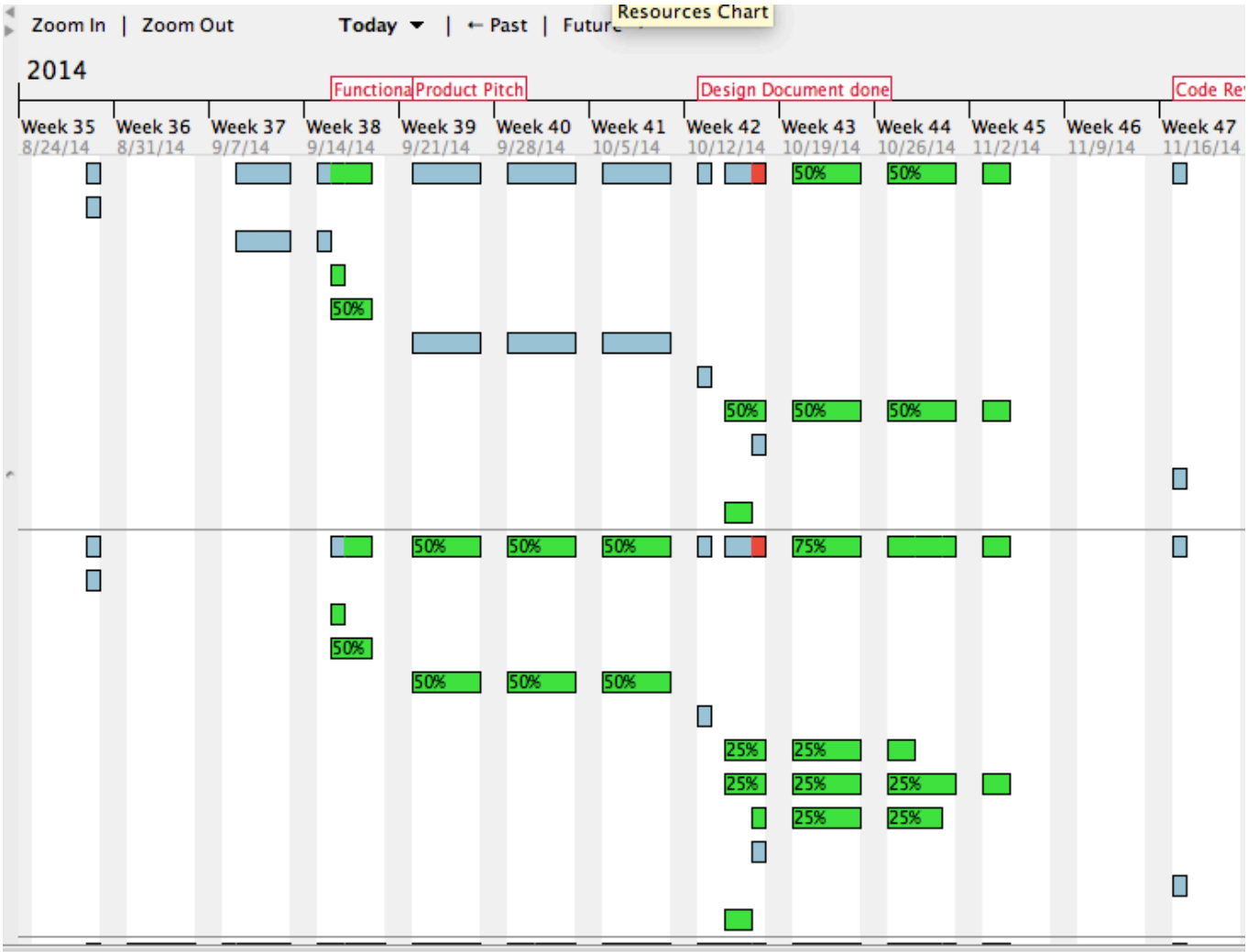


Table	Description
Users	The Users table is the table that contains all user information. This table has the following attributes: userID, userName, firstName, lastName, Password, and email. The users userID is the key that will be used to represent that user. When a user leaves a review their userID will be associated with that review. The primary key for this table will be the userID, userName, and email. Eventually we will implement foreign keys that will be related to the reviewFor, reviewBy, ratingsFor, ratingsBy attributes from the Reviews and Ratings table respectively. These data values will be filled in when a user creates an account with the application.
Reviews	The Reviews table is the table that will contain all of the reviews that users have written. The Reviews table has the following attributes: reviewFor, reviewBy, date, text, class, and school. The reviewFor and reviewBy attributes correspond to the userID of that user. For example, if userX leaves a review for userY then reviewFor will be userY's userID, and reviewBy will be userX's userID. As of right now this table does not have a primary key, we are thinking about adding in a reviewsID so that we can distinguish keys, but have to discuss this further. We did not implement a primary key because we are under the assumption that a user can leave multiple reviews for the same user and they should not be restricted. The data values for this table will be populated when a user is leaving a review for a mother student.
Ratings	The Rating table is the table that will contain all the ratings given to a user. The Ratings table has the following attributes: ratingFor, ratingBy, knowledge, availability, motivation, dependability, and friendliness. ratingsFor and ratingsBy serve the same functionality as the reviewFor and reviewBy attributes in the Reviews table. The other attributes will be integer values between 0-5 or 1-5, the values will be added together and then the average will be take to determine the users average rating based on all ratings that have been left for that person. The primary key for this table will be the ratingFor and ratingBy attribute. We did this because we wanted to restrict a users rating of another classmate to one. This data will again come from the user filling out a review for a classmate.
Schools	The Schools table is the table that will contain all of the schools in the United States. The Schools table has the following attributes: schoolID, name, city, state. We have found a CSV file that contains all of the schools and we will use this to populate the database, Note what we have not implemented this yet. A user will be able to select the list of schools from a drop down menu or through a search function. This will be to prevent users from entering fake schools. The primary key for this table is the schoolID, in the CSV file that we found it contains unique ID numbers for every school.


Zoom In | Zoom Out
Today | ← Past | Future → | Show critical path | Baselines...
2014
Name | Begin date | End date
RateMyClassmate 8/29/14 12/10/14
Get requirements 8/29/14 9/15/14
Brainstorm Product 8/29/14 8/29/14
Create use cases 9/1/14 9/8/14
Write functional s... 9/9/14 9/15/14
Functional specs... 9/16/14 9/16/14
Design 9/16/14 10/10/14
Conceptual Design 9/16/14 9/18/14
Prototype 9/16/14 9/19/14
Product Pitch 9/22/14 9/22/14
Write Design Spec 9/22/14 10/10/14
Design Documen... 10/13/14 10/13/14
Coding 10/1/14 12/10/14
Iteration 1 10/1/14 10/14/14
Database 10/1/14 10/14/14
Iteration 2 10/15/14 11/4/14
Database 10/15/14 10/28/14
Servlets 10/15/14 11/4/14
JSP 10/15/14 11/4/14
Regression T... 10/15/14 10/16/14
Design Review 10/17/14 10/17/14
Iteration 3 10/17/14 11/14/14
Servlets 10/17/14 10/30/14
Code Review 11/17/14 11/17/14
Testing 11/17/14 12/10/14
Product Demo 12/11/14 12/11/14
Week 35 8/24/14
Week 36 8/31/14
Week 37 9/7/14
Week 38 9/14/14
Week 39 9/21/14
Week 40 9/28/14
Week 41 10/5/14
Week 42 10/12/14
Week 43 10/19/14
Week 44 10/26/14
Week 45 11/2/14
Week 46 11/9/14
Week 47 11/16/14
Week 48 11/23/14
Week 49 11/30/14
Week 50 12/7/14
Product Pitch
Design Document done
Code Review
{Brandon},Arash,Cody,Nabil
{Nabil},Cody
{Brandon},Cody
{Brandon},Arash,Cody,Nabil
{Brandon},Arash
{Cody},Nabil
Brandon,Arash,Cody
{Nabil},Brandon,Arash,Cody
{Cody},Nabil
{Arash},Nabil
{Nabil},Arash
{Brandon},Cody
{Arash},Brandon
{Brandon},Arash,Cody,Nabil
{Cody},Arash
{Brandon},Arash,Cody,Nabil


Zoom In | Zoom Out
Today | ← Past | Future
Resources Chart
2014
Product Pitch
Design Document done
Week 35 8/24/14
Week 36 8/31/14
Week 37 9/7/14
Week 38 9/14/14
Week 39 9/21/14
Week 40 9/28/14
Week 41 10/5/14
Week 42 10/12/14
Week 43 10/19/14
Week 44 10/26/14
Week 45 11/2/14
Week 46 11/9/14
Week 47 11/16/14
50%
75%
25%

Release Notes

REQ-2: USER may browse complete list of user rating profiles

REQ-5: USER has ability to validate other reviews with upvote/downvote system.

NRF-2: USER profile password must be no less than seven (7) characters long

These features may be implemented in future versions.