Web Programming – Final Project

Winter 2015-16 CS@Haifa University

Course Instructor: Dr. Haggai Roitman, IBM Research - Haifa

Publication date: 1/1/2016, (HARD) Deadline: 22/2/2016, Version: 1.2

Project scope: explore, design and implement a web application using the various client-side and server-side technologies that were thought during this semester.

Project general description: the goal of this project is to implement a **Community Question-Answering (CQA)** service. The service will allow users to submit questions on any possible topic or answer questions of others. Example real-life CQA services include Yahoo! Answers, Quora, StackOverflow, etc. User contributed questions or answers may be rated by the members of the community. The service shall allow users to explore new questions that have not been answered yet, so as to allow others to answer. The service will further allow users to track previously posted questions and the answers that were given to them by members of the community. The service shall maintain a rating for each user, representing that user's reputation. A user's reputation will be based on the overall ratings given by the other members of the community to her contributed content (i.e., questions and answers).

Outline

The followings describe the project structure (requirements, guidelines, tips, etc.).

- **Functional requirements** are a set of features that you are expected to implement during the project. **Please make sure you have covered all requirements.**
- **Implementation requirements** are a set of requirements that define what web technologies you are allowed to use and which are a must. In addition, general development guidelines are provided. Please pay attention.
- **Design and UX requirements** are a set of requirements about the UI design of your application and its user experience. Please pay attention.
- **Deployment and Environment requirements** are a set of requirements for properly preparing your project for submission. Please pay attention.
- **Bonus requirements** are a set of requirements that you need to fulfill so as to deserve an extra credit for your project (**up to 10 points can be earned in final grade! Final project grade can't be higher than 100, though** ☺).
  Project submissions that will not fulfill the basic requirements will not be given a bonus. Therefore, don't waste your precious time before you finish those requirements.
- **Submission guidelines** are a set of guidelines of what should be submitted, in what format, using what naming convention and where to submit, etc**.** Please pay attention.
- **Tips** are a set of guidelines that should "ease" your life. Such tips are partially given throughout this document, but some others may pop-up as you shall work on this project. Don't ignore those tips, keep tracking for updates at class or in the course online forum.
- **Grading guidelines** is the approximate grading scheme that will be used to judge the quality of your work in this project. Please pay attention.

- **Project integrity guidelines** is a set of integrity rules about this project; basically, a list of "do and not do" (especially to make sure your work is genuine). **Integrity guidelines will be strictly enforced**. Please pay attention.

Functional requirements:

The following is the list functional requirements that your CQA application must support **(use this requirements list as your check-list; make sure you have covered each one and one of these requirements!)**

a. A user should log into the system using a username and a password
b. In case the user doesn't exist, the user should be able to register to the service.
    i. The user will provide the following details during registration:
        1. **Username** (required and unique, up to 10 characters) – username represents the user internally in the system.
        2. **Password** (required, up to 8 characters)
        3. **User nickname** (required, up to 20 characters) – nickname is a public name that will be displayed to other users in the system.
        4. **Short description** (optional, up to 50 characters).
        5. **Photo** (optional, size 50x50) – link to photo should be provided by **URL**. This photo will be linked in the application.
           No need to support image upload from the user file-system!
    ii. Upon registration, the user will be shown the main application's UI.
c. Registered users will be shown the main application's UI.
d. The main UI will allow the user to perform the following operations:
    i. Question submission
        1. The user will be able to submit a new question to be answered by other users of the community.
        2. For each question, the following details should be provided:
            a. **Question's text** (required, up to 300 characters)
            b. **Question's topic(s)** (optional, a list of tags up to 50 characters, each)
        3. Upon submission the new added question should be visible in the list of "**newly submitted questions**" (see next requirement).
    ii. View newly submitted questions
        1. The user will be able to see a list of newly submitted questions. A question is new if no user has provided an answer to it yet.
        2. A newly added question, submitted by some user will be **immediately** displayed in this list.
        3. Questions will be displayed in chronological order (i.e., the most newly added questions should appear first).
        4. The list will display the 20 newest added questions.
        5. The user will be able to browse through the list (i.e., move forward to the next 20 questions or backward in the list).
        6. For each question, the following details will be displayed:
            a. **Time of question submission** (in format dd/mm/yyyy hh:mm:ss)
            b. **Question's text**

c. **Question's topic(s)** (a list of tags up to 50 characters, each)
d. **Nickname of user** that submitted the question
e. **Question's rating** (should be 0 since this is a new question).

7. The user will be able to select a question to answer.
8. User's answer should appear bellow the question's text.
9. User's answer will include the same details as questions:
    a. **Time of answer submission** (in format dd/mm/yyyy hh:mm:ss)
    b. **Answer's text** (up to 300 characters)
    c. **Nickname of user** that submitted the answer
    d. **Answer's rating** (initialized as 0 since this is a new answer).
10. The user will be able to vote any question, either give a positive vote (denoting the user thinks the question is important) or negative (denoting the user thinks the question is of no importance).
11. Upon user vote to a given question (either positive or negative), its rating score will be updated (either +1 if positively voted or -1 else).
12. A user can't vote for her own questions or answers.

iii. Browse existing questions

1. Questions will be displayed in a list in a similar fashion to new questions (i.e., up to 20 displayed questions, further allowing the user to browse forward/backward through the list).
2. For each question, its answers (if exist) will be displayed to the user in a "thread" fashion.
3. Answers for a given question will be displayed according to the answer votes (i.e., an answer that got more votes will be displayed first).
4. For each question, only the highest voted answer should be displayed, while the rest of the answer thread should be displayed only upon user's request (e.g., the thread is expanded for display after user clicks a button to expand the list).
5. Questions will be displayed according to their rating score (in descending order).
    a. A question's rating is determined by the following formula:
    ```
    QRating = 0.2 * (Question own voting score) +
    0.8 * (Average voting scores for answers given
    to the question by far)
    ```
6. The user will be able to select a question to answer.
7. The new answer will be added to the question in a thread fashion as follows:
    a. If this is the first answer given to the question, then it will be added bellow the question (see the required answer details as in requirement [**ii.9**])
    b. If the question already has one or more answers, then the new answer will be added to the bottom of the list of answers in a "thread" fashion.
8. The user will be able to rate (vote) any question or answer that appears in the list.
9. Questions rating by users will be implemented similarly to requirements [**ii.10-12**].
10. The user will be able to provide a vote to any question's answer in a similar fashion to questions votes (see requirements [**ii.10-12**]).

iv. View users' leaderboard

1. The user leaderboard will display the top-20 rated CQA users (in descending order).
2. For each user in the list the following details will be displayed:
    a. **User's photo**
    b. **User's nickname**
    c. **User's rating**
    d. **User's expertise profile: a list of the top 5 topics of user**
        i. Each topic denotes some expertise area of a user and its "rank" for the user is determine by the overall number of votes of answers that belong to questions in this topic that the user has answered by far.
3. A user's rating is determined by the following formula: `URating = 0.2 * (Average rating of user's submitted questions) + 0.8 * (Average rating of user's submitted answers).`
4. The current user will be able to view a summary of each user displayed in the leaderboard by clicking a given user's nickname
5. Upon click on some user's nickname, the following summary will be displayed:
    a. **User's photo**
    b. **User's nickname**
    c. **User's short description**
    d. **User's rating**
    e. **User's expertise profile** (see requirement [**iv.d.i**])
    f. **User's 5 last asked questions,** where for each question the following details will be further displayed:
        i. **Timestamp** (see requirement [**ii.6.a**] for format)
        ii. **Question's text**
        iii. **Question's topics**
        iv. **Question's rating**
    g. **User's 5 last answered questions and the user's answer**. Each question's details will be as in requirement [**iv.5.f**]. In addition, the user's answer text and its answer rating will be displayed bellow the associated question.

v. Browse questions by topics

1. The user will see a list of the top-20 most popular topics.
2. Topic popularity is defined by the following formula:
   `TPop = Sum over QRating of queries in this topic.`
3. The user will be able to browse forward and backward to see more topics (in order of topic popularity). Alternatively, you may wish to implement the topic browsing feature as a tag cloud (see https://en.wikipedia.org/wiki/Tag_cloud).
4. The user will be able to click on any topic from the list and see the list of top-20 highly rated questions in that topic similarly to the way questions are displayed in requirement [**ii**]

Implementation requirements:

1. You are only allowed to use web technologies that were taught during the semester.
2. You should use HTML 5 (with UTF-8 encoding) and CSS 3 (most browsers support both). You are allowed to use any advanced HTML 5 or CSS 3 features as you wish.
3. Keep correct usage of the technologies, i.e., content should be given in HTML, style and layout in CSS (or Bootstrap), logic should be located in JavaScript, Servlets, data in the database, etc.
4. Don't define inline or embedded style rules. Instead, define such rules in a linked CSS file.
5. You are not allowed to use any third-party (or open source) libraries of any kind for the server side code; except those required by Eclipse Mars, Tomcat and Java Servlets (the later should be included in Eclipse Mars J2E distributions).
6. You are not allowed to use any third-party (or open source) libraries of any kind; especially JavaScript libraries (except for the **basic** jQuery, Bootstrap and AngularJS libraries).
7. You are not allowed to use any other J2E server-side technology except for regular Java code, Servlets and JDBC features (e.g., don't use JSP, EJB, JSP-Tags, etc.).
8. You must use **Java Servlets version 2.5**. **Be aware of more recent versions that auto-generate J2E Annotations!**
9. All communication with the server should be done using a **RESTful** API.
   POST messages should contain parameters sent to the server within a JSON format.
10. Data transferred from the server to the client and backwards must be in JSON format. For server side JSON java utilities use: https://github.com/google/gson
11. You should only use the Apache Derby open source Database.
12. Use proper coding styles:
    a. Follow Java code conventions (see http://www.oracle.com/technetwork/java/codeconventions-135099.html)
    b. Follow JavaSscript code conventions (see http://javascript.crockford.com/code.html)
    c. On the client side, document with `<!--comments -->` your HTML code whenever needed for clarity.
    d. Use //comments in JavaScript to document your code
    e. Use /*comments*/ in CSS to document your style rules
    f. Use Javadoc documentation guidelines:
        i. Use /**….*/ for public (user) comments
        ii. Use /*…*/ or //… comments for developer comments
        iii. Document method input parameters (*@param*) , method outputs (*@return*), exceptions (*@throws*), and dependencies (*@see*)

Design and UX requirements:

1. Your client-side UI should be completely developed with Bootstrap.
2. UI must be responsive.
3. You are NOT allowed to use any additional (third-party or open source) UI features (e.g., jQuery-UI widgets).
4. You are NOT allowed to use ready templates. **Usage of such a template will be easy to trace, and your project will be disqualified**.
5. **Usability**: make sure your application correctly follows the usability guidelines we have learned during the semester (e.g., make sure images have <alt> tags; fonts, colors, etc. are properly chosen, etc.).

6. **You are more than encouraged to use your own creativity in designing your application and its theme.** For example, your CQA implementation may focus on specific domain (e.g., lifestyle, foods & cooking, etc.).

Deployment and Environment requirements:

1. The following third-party/open source libraries should be used in your project for development and deployment. **Do not use any other code/software packages etc.**
    a. Java 7 JRE http://www.oracle.com/technetwork/java/javase/downloads/jre7-downloads-1880261.html
    b. Eclipse IDE (Mars) for J2E: https://eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/mars1
    c. Apache Tomcat v.7: http://tomcat.apache.org/download-70.cgi
    d. Bootstrap: http://getbootstrap.com/getting-started/#download
    e. jQuery: http://code.jquery.com/jquery-1.11.3.min.js
    f. AngularJS: https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0-beta.2/angular.min.js
    g. Apache Derby: http://db.apache.org/derby/releases/release-10.12.1.1.cgi
2. You may want to use the JUnit tool for testing your java code: http://help.eclipse.org/luna/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2FgettingStarted%2Fqs-junit.htm
3. **Make sure your application pass W3C Validation**: http://validator.w3.org/

Bonus requirements:

The following are four options to earn extra credit to the final project grade. You may choose to implement **up to two out of four** such options.

1. **[5 bonus points] Entity Search:** support textual search feature over questions, answers, topics and users. You must use the Apache Lucene open source search library for this purpose: http://www.apache.org/dyn/closer.lua/lucene/java/5.4.0
2. **[5 bonus points] Visualization:** using the Data-Driven Documents (D3) framework (http://d3js.org/) implement a visualization that can be useful for users in such a CQA setting using the current application you have developed.
3. **[5 bonus points] Personalization**: using the Apache Mahout TASTE package, implement a simple personalization service (e.g., recommender system) for your application. For example, such service may be used to recommend newly unanswered questions to users who may wish to answer.
   https://mahout.apache.org/users/recommender/recommender-documentation.html
4. **[5 bonus points] Creativity track:** you are more than encouraged to suggest a cool feature of your own! Be creative!

Project submission guidelines:

Due date (HARD DEADLINE - NO EXTENSIONS!): 20/2/2016

You will need to submit the following material enclosed inside a directory named **"webappproject-ID1-ID2**", where ID1 and ID2 are the student ids of the submitting team. Further zip your submission (using only **.7z** or **.zip** compressions!).

Your submission should be sent to haggair@gmail.com with the email title: "*web proj. sub.*" Submissions will get confirmation from me upon receive. **You may submit only once. Hence, please make sure your applications correctly runs.**
I will not try to fix your submissions.

1. **Students.txt**: will include details of your **ids** and **names**.
2. **ERDdiagram.ppt:** a power-point (or image file) that will include the ERD model of the data entities and relationships of your database.
    a. **Please keep the rules of ERD that were taught in class.**
3. **DBSchema.sql**: all SQL DDL (i.e., CREATE TABLE…) and SQL DML (i.e., SELECT …) commands that you have used for creating and manipulating your Derby database. Please document any assumptions you have made on the data.
4. **webapp.zip:** the archive file (in zip format only) of your project that was developed in Eclipse (**use Eclipse export utility for this!**).
    a. **Java code should be well documented using Javadoc commands** (i.e., General description of classes, methods, @param, @return, @throws, etc. No need to document getters and setters.)
    b. Your submitted package should contain all resources (i.e., static web pages, .js, .css, .class, web.xml, Tomcat's context.xml) files that your web project depends on for proper running.
5. **javadoc.zip:** the Javadoc of your project. The following instructions show how to generate a Javadoc for your project in Eclipse: http://help.eclipse.org/juno/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Freference%2Fref-export-javadoc.htm

Project grading:
The following grading guidelines will be used to evaluate your project (max project grade is 100):

1. Requirement fulfillment: 70-75%
2. UI and UEX (Usability): 10%
3. Documentation: 5%
4. Proper coding style: 5%
5. Creativity: 5-10%

Project integrity guidelines:

1. Project must be prepared **in pairs only**.
2. Teams should not copy or share code between each other.
3. Code or template usage from third-parties or open source is completely forbidden.
4. **Failing to follow these basic guidelines will disqualify your project**.

Last and for all:

The purpose of this project is to get real experience with what we have learned. This can be only done through "diving into the deep water". If you get stuck, have questions, etc., try to utilize the following resources:

1. Online communities: I warmly recommend to post questions/look for answers in real CQA websites ☺ such as on StackExchange, CodeAcademy and Quora.
2. General topics may be discussed in the course forum, **though you are not allowed to share code through this medium**.

3. I will devote up to 15 minutes at the beginning of every lecture to discuss issues related to the project.
4. If you had some important assumptions you have made during the development, document them, and when you are not sure, just ask me for confirmation.

GOOD LUCK!