Web Programming – Final Project

Winter 2017-18 CS@University of Haifa

Course Instructor: Dr.Guy Feigenblat, IBM Research AI

Teaching Assistant: Mr. Sami Marid, University of Haifa

Publication date: 11/12/2017, (HARD) Deadline: 25/2/2018- 08:00:00, Version: 1.0

- Based on a project definition template by Dr. Haggai Roitman -

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

Project general description: the goal of this project is to design and develop **e-commerce** web application (hereinafter termed the "*BooksForAll*"). The application allows users to buy e-books, read them on their browser, and share their feedback with respect to books they have read.

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 application must support **(use this requirements list as your check-list; make sure you have covered each one and one of these requirements!)**

Regular User - Functional Requirements:

✔ **1. Login, registration, landing page**
    a. A user should log into the system using a username and a password
    b. In case it is a first login, the user should be able to register to the application. 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. **Email** (make sure email is valid)
        3. **Address** (make sure address is valid , use Israel Mailing Address Formats)
        4. **Telephone number** (make sure phone number is valid- mobile, landline)
        5. **Password** (required, up to 8 characters)
        6. **User nickname** (required, up to 20 characters) – nickname is a public name that will be displayed to other users in the system.
        7. **Short description** (optional, up to 50 characters).
        8. **Photo** (optional, size 50X50px) – link to a 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!**

    c. Upon registration, user logs in to the main page of the application.
    d. The landing page (welcome page) of the application <u>must</u> display the logo of the company, the name of the logged in user and links to other functionalities available to the user. With respect to its design think creative. You can get your inspiration from various well known e-commerce applications.

✔ **2. Browse e-books available for purchase**
    a. User should be able to browse all available e-books
    b. The UI should display for each e-book the following details :
        i. Name
        ii. Image
        iii. Price
        iv. Short description
        v. Number of "like"s (with a tool tip showing nick names of users who liked the book)
        vi. Collapsed list of reviews
    c. For better usability the UI should support scrolling
    d. See (6) with requirements with respect to how to buy an e-book.

✔ **3. Browse reviews of e-book**
    a. Upon selecting e-book, user has the ability to read verified reviews of e-books
    b. For better usability the UI should support scrolling

✔ **4. Browse my e-books**
    a. User can select to browse over the list of her purchased e-books.
    b. Similar to (2) the user should have the ability to read verified reviews of e-books
    c. For better usability the UI should support scrolling

5. **Read e-book**
   a. Upon selection of e-book from the my e-books list, the e-book is opened for reading
   b. User read the e-book online
   c. The location of the scroll is saved
   d. If user has already started to read the e-book she is offered to jump to the position where she last stopped

6. **Buy e-book**
   a. User can choose to buy e-book
   b. Upon selection the user is asked to "pay" for it
   c. User specifies his billing information
   d. User selects her credit card company (e.g. VISA, AMEX etc).
   e. User enters credit card information- number, expiry, CVV, full name (make sure to support various formats)
   f. Payment verification - verifying that the user has entered valid information on (c)-(e)
   g. Once payment is approved the book is added to the user's "my e-books" list in the application

7. **Write a public review on a book**
   a. User has the option to write a public review on a book she has purchased (underline verified purchase)
   b. Review is publically visible only after administrator user has approved it

8. **Like e-book**
   a. User can choose to "like" a book she has purchased
   b. The "like" is added to the overall number of likes a book has received

9. **Unlike a book**
   a. Users can revert their "like" selection.
   b. The "like" should be properly removed from the overall "like"s a books has received

Admin User - Functional Requirements:

1. **Login, Landing page**
   a. Login and landing page design is similar to regular user. No need to support registration of Admin

2. **Browse users**
   a. Admin should be able to view all users
   b. The component should display all user details in a usable format (think how)

3. **Browse list of e-books**
   a. Admin can browse the list of e-books available in the application
   b. The component should display the following details for each book:
      i. Name
      ii. Image
      iii. Price
      iv. Short description
      v. Number of likes
         1. With a tool tip showing nick names of users who liked the book

2. Nick name should be clickable
3. Selecting a nick name takes to user details page
vi. Collapsed list of reviews (can be expanded on selection)

4. **Remove User:**
   a. Admin can remove users
   b. Confirm message should be shown before the task is performed

5. **View recent e-books transactions**
   a. Admin user can review the history of e-books purchases in the application
   b. A table should be displayed with this information (think about usability)

6. **Approve a review**
   a. Before review is publicly available admin should approve the review
   b. The application should display all un-approved reviews
   c. Admin read an un-approved review and selects whether to approve it
   d. Review should only be written by a user who purchased the e-book

General Requirements:

1. Some of functionalities are designated for regular users and some for admins (two roles are defined). Make sure you confirm the user identity before granting access to a functionality (hint think about filter servlets).
2. Pre-configure one administration account:
   a. Username: admin
   b. Password: Passw0rd (with the number "0" instead of the letter "o").
3. Use the Gutenberg project (https://www.gutenberg.org/) to download several e-books (about 10) in HTML format and populate on your application. You aren't required to support adding new e-books dynamically. Only the HTML should be stored inside the war as a static content and **not** in the database (in a folder under the *WebContent* of your dynamic web-project).
4. Populate your application, with 20 users, about 10 reviews for each book, and several "likes". For the reviews you are allowed to utilize any public dataset of reviews even if the review isn't related to the actual e-book (such as http://www.cs.cornell.edu/people/pabo/movie-review-data/). Each review should be longer than 4 sentences.

Implementation requirements:

1. **You are only allowed and <u>required</u> 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).
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, or Endpoints; 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 Oxygen, Tomcat and Java Servlets (the later should be included in Eclipse Oxygen 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, Endpoints and JDBC features (e.g., don't use JSP, EJB, JSP-Tags, php, etc.).
8. You must use **Java Servlets version 3.0**
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 (e.g., no MySQL!!).
12. Make sure user input is carefully validated twice- before submission and on the server.
13. 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 and adaptive to various screen sizes.
3. You are NOT allowed to use any additional (third-party or open source) UI features (e.g., jQuery-UI widgets, Dojo, etc).
4. You are NOT allowed to use readymade 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: specifically think about scrolling, tooltip sizes, sorting (ascending \ descending) etc.
6. **You are more than encouraged to use your own creativity in designing your application and its theme.**

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 8 JJDK http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html
    b. Eclipse IDE (Oxygen) for J2E http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/oxygen1a
    c. Apache Tomcat v.8.5: http://apache.mirrors.pair.com/tomcat/tomcat-8/v8.5.23/bin/apache-tomcat-8.5.23.exe
    d. Bootstrap: http://getbootstrap.com/docs/3.3/getting-started/#download
    e. jQuery 3.2.1 : http://code.jquery.com/jquery-3.2.1.js

      f. AngularJS 1.6.6 : https://ajax.googleapis.com/ajax/libs/angularjs/1.6.6/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/

4. **All your third-party/open source libraries should be delivered with your project (JavaScripts under web-content and jars under WEB-INF/lib. Assume we will test the project on an environment with no internet connection. Make sure you carefully test that.**

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. **[up to 5 bonus points] Search Service:** support textual search feature over books and reviews. **You must use the Apache Lucene 6 open source search library for this purpose.**

2. **[up to 5 bonus points] HelpMe:** write an internal messaging service that will enable users to send messages to admins and for admins to respond (think creatively!).

3. **[up to 5 bonus points] Visualization:** using the Data-Driven Documents (D3) framework (http://d3js.org/) implement a visualization that can be useful for admin users using the current application you have developed.

4. **[up to 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!): 25/2/2018- 08:00:00 (am)

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 guyf.web.programming@gmail.com with the email title: "*web proj. sub.*" Submissions will get confirmation from us upon receival. **You may submit only once. Hence, please make sure your applications correctly runs. We will not try to fix your submissions.**
Please submit the following parts and <u>keep the EXACT filenames!</u>

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. **BooksForAll.war:** the war distribution of your project. The war should include all static content, class files and java source files. Make sure target runtime is optimized for ApacheTomcat V8.5. (**use Eclipse export utility for this!**).
6. **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
7. **Link to your project in GitHub repository**

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 UX (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: we warmly recommend to post questions/look for answers in community question answering websites such as 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. We may devote up to 15 minutes at the beginning of every lecture/tutorial 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 us for confirmation.

GOOD LUCK!