**Inventory Review Final Project**

**This final project is copyrighted and may not be shared or posted online in any format! The solution, written by you, is a derivative work of this project and is also copyrighted. It may not be shared, posted, or made available in any form, other than to your GitHub repository and production server for the purposes of this class.**

This project is intended to stretch you and be a thorough review of all concepts introduced in the course. You are encouraged to work with your learning team to finish the project, but all work must be your own. No late submissions will be accepted!

**Project Tasks**

You are to build an application that will allow inventory item reviews to be added to the site. Specifically:

1. Use an MVC approach, including appropriate routes, for the solution.
2. Build the “review” database table along with appropriate relationships to the inventory and account tables in the existing database. Store the SQL code to the database folder in the project.

|  |  |  |  |
| --- | --- | --- | --- |
| Column name | Column default | Null? | Data Type |
| review\_id |  | no | Integer, generated as identity |
| review\_text |  | no | text |
| review\_date | now() | no | timestamp with timezone |
| inv\_id |  | no | integer |
| account\_id |  | no | integer |

1. The needed processes are:
   1. Add a new review (form in the detail view as described below).
   2. Edit and delete an existing review (limit to the author, add views as needed).
   3. Deliver reviews to the "detail" view along with the existing inventory item details. Within the detail view, include the following functionality:
   4. Create a clear "Reviews" heading after the content area, but before the footer.
   5. Beneath the "Reviews" heading show existing reviews for the item. Querying for existing reviews will have to become part of the process to build the detail view.
   6. Reviews that are displayed in the vehicle detail view must show the most recent review listed first and the oldest review listed last.
   7. Individual reviews must include 1) the review text, 2) the reviewer's screen name
   8. Beneath any existing reviews, add text indicating that a review can be added by "logging in" and provide a link to deliver the "login" view. If the client is already logged in, then provide the form for entering a review. The form must:
      1. Only provide a text area for the review to be written.
      2. Display the "screen name" (the first initial of the first name and the complete last name, with no spaces) in the form and not be editable.
      3. Include the inventory ID in a hidden field in the form.
      4. Include the account ID in a hidden field in the form.
2. In the existing "Account Admin" view display a list of reviews (if any) that the logged-in client has written with the ability to update or delete the individual review.
3. If the client opts to update a review, the review information must be displayed in the update form within a review update view for editing. Only the review text should be editable! Implement appropriate client and server-side validations for updated data.
4. When a review update or delete is finished, the "Client Admin" view should be delivered with an appropriate message indicating the outcome of the process.

**Submitting**

When done testing and satisfied that everything is working as it should, do the following:

1. Deploy the working code to your render.com site.
2. Ensure that the SQL code used to create the review table exists in the “database” folder.
3. Create a zip file of the CSE Motors application folder.
4. Submit the zipped application folder to the code submission dropbox and paste the render.com production URL to the code submission comment (just as you have done all semester).

**Grading Matrix**

**Objective 1**

* All views meet the requirements of the frontend checklist.
* Views are laid out in a professional, useable manner.
* Form inputs are sticky when server-side validation errors arise.
* Success and error messages are styled and easy to discern.

**Objective 2**

* Appropriate routes exist to direct review-related processes to the correct controller.
* Inventory controller operates to deliver reviews with the inventory detail content.
* Account controller operates to deliver client reviews upon log-in, for editing and deleting.
* Review controller operates to add, update, and delete reviews.

**Objective 3**

* An MVC solution has been created and works as expected.
* Middleware limits access to review submission forms to logged-in clients and works correctly.
* Controllers contains all needed logic and function correctly.
* Model contains all database interactions, returns results to controller, and works correctly.

**Objective 4**

* Review table was created and stores all reviews for the application correctly.
* Review functions in the model use a prepared statement approach and function correctly.

**Objective 5**

* Client-side and server-side validation is present and functions correctly.
* Errors, when detected, are returned, and displayed correctly.

**Objective 6**

* The project zip file and production URL are complete and delivered on time