Assignment: Mastery, Car Dealership

Re-submit Assignment

Due No Due Date **Points** 100 **Submitting** a website url

Overview

Project source code: [Car-Dealership-Wireframes.pdf (https://lms.thesoftwareguild.com/courses/424/files/79020/download?wrap=1)](https://lms.thesoftwareguild.com/courses/424/files/79020/download?wrap=1) [(https://lms.thesoftwareguild.com/courses/424/files/79020/download?download_frd=1)](https://lms.thesoftwareguild.com/courses/424/files/79020/download?download_frd=1)

Congratulations on making it this far! We have come a long way on our journey to entry level developer skills and it is time to prove our mastery by creating a full-stack web application that leverages the competencies covered in all sections of our program.

The application we would like you to create models a simplified version of a car dealership's website and sales support system. Like many car dealership sites, this application provides a public-facing area which allows users to browse the inventory of new and used vehicles, view current specials, and contact the dealership. Additionally, for sales personnel, there is an area for logging customer information for a purchase and an administrative area for user management, application management, inventory management, and reporting.

Your assignment is to complete and submit a working full-stack application. There are two parts to turn in for this assessment: a database build script and the web application code.

Part 1 – Database Build Script

You must provide a series of SQL scripts to build the car dealership's database schema and supporting objects.

- A database.sql file should contain the scripts to drop and create the database, which should be named GuildCars.
- A tables.sql file should contain the scripts to drop and create the tables for the database as well as configure their foreign key relationships.
- Upon completing these scripts you should submit a code review request. A staff member will review your tables and relationships to ensure:
    - The necessary fields for the web application are present with appropriate types and nullable settings.
    - Tables are in at minimum second normal form (2NF).

Note that in this code review we do not promise to catch all errors and omissions. The purpose of the initial code review is to ensure that your relational database structure will reasonably address the requirements of the application and demonstrates competency in relational database architecture. You ~~may~~ **will** need to make minor changes to your data model when building the application.

Part 2 – Web Application

Your solution should be properly layered and contain a data layer project, a models project, a unit test project, and the UI project.

Data Layer

The data layer should be designed according to principles learned in the program:

- You may use ADO.NET, Entity Framework, or Dapper for the repository classes. If you choose Dapper or Entity Framework you must provide at least one example of an ADO.NET call that requires the DataReader class with a stored procedure call.

- Each repository should implement an interface for dependency injection via Factory classes or 3rd party solutions like Ninject or Unity.
- You should provide an in-memory and database enabled version of each repository configurable via an <appSettings> entry named “Mode”. The mode values should be “QA” and “PROD”.
- Database access should be done via Stored Procedures which should have create and drop scripts in a “sprocs.sql” file. The exception to this is search queries, which may be written with in-line SQL.

Models Layer

The models layer should be designed according to principles learned in the program:

- This layer should contain C# model classes that are used by the repositories.
- This layer should **not** contain data annotations or any other UI logic or validation.

Unit Tests

You must provide integration tests for your data layer.

- Please provide integration tests for the database using a reset script to provide a consistent database state in-between tests.
- NUnit should be the testing framework utilized for this project

UI Layer

Create an ASP.NET MVC 5 project with Web API enabled.

- Use the Empty template to start the project.
- Install and configure ASP.NET Identity for authentication and authorization.
- Install the latest version of bootstrap, jQuery, and jQuery validate.

JavaScript and CSS frameworks that were not part of the course materials are not permitted in this project.

- Be sure to maintain the URL paths provided in the specifications below

Wireframes

Refer to the Wireframes.pdf file for the business' vision for this application. Keep in mind that specifications in the real world are rarely complete or 100% correct, so treat your assigned instructor as the business owner of this project and be sure to direct any clarifying questions about the specifications to them. Utilize any Guild staff member for technical assistance.

The provided wireframes should be used to intuit the database structure.

Home/Index

This is the entry point for the public to the application.

- A standard bootstrap navigation bar should appear with a placeholder logo.
- A bootstrap jumbotron should appear as the main banner for this page.
  - This banner should have a background image. This image may be a solid/plain color.
  - Specials information should be stored in the database. Each special that is present in the database should be rotated through the jumbotron. Thus, if there are 5 specials in the database, the jumbotron should rotate through the 5 specials.
  - Clicking on a special should send the user to the specials page (home/specials).
- A short list of featured vehicles should appear below the specials jumbotron.
  - Vehicles only show up in this area if they are marked as featured in the admin/editvehicle page.

Inventory/New

This allows users to search the new vehicle inventory of the site.

- No search results should be shown by default
- Searches should be dynamic, such that if a filter is not provided (ex: choosing “No Min” on price) then minimum price should not be a factor in the search.
- The “make, model, or year” textbox, when filled in, should perform a LIKE query that searches all 3 columns in the database. (ex: ‘fo’ should match all ‘Ford’ makes as well as a Subaru ‘Forrester’)
- The search should utilize an AJAX call via jQuery.
- Only the first 20 matches should be returned.
- Leaving the filters blank and hitting search should return the 20 vehicles with the highest MSRP.
- Only vehicles with a type of ‘New’ should be returned in this search.

Inventory/Used

Use the same rules and logic from the Inventory/New page excepting that this search should only return vehicles with a type of ‘Used’.

Inventory/Details/id

This page should load the vehicle information with the appropriate database id.

- The “Contact Us” button should automatically put the VIN # into the message text box of the contact page.
- If the vehicle has been purchased, the “Contact Us” button should be removed and replaced with a label that reads “Unavailable- Sold”

Home/Specials

This is a list of all the specials information from the database. An administrator deletes specials when they are no longer needed.

Home/Contact

This form should put a contact record in the database.

- Name and message are required.
- If the contact button on the Inventory/Details page was used to navigate here, the VIN # of the vehicle should be placed into the message field.
- Either email or phone must be provided. Leaving both empty should cause a validation error.
- Embed a Google map to the address of the dealership. You may make up any address you like for the dealership for this assignment. Instructions to embed a Google map can be found on the Google Maps documentation site.

Sales/Index

This page should only be accessible to users in the “Sales” role.

- This search should behave the same as the New and Used inventory searches with two exceptions:
  - It searches both New and Used inventory simultaneously.
  - A purchase button is listed on each result.
  - Vehicles that have already been purchased should not show up in search results.
- The search should utilize a jQuery AJAX call.

Sales/Purchase/id

This page allows a salesperson to log a purchase. It is only accessible to users in the “Sales” role.

- Either phone or email (or both) must be present to complete a purchase.
- Street 2 is optional.
- All other fields are required.
- Zip code must be 5 digits.
- Email must be a valid email format if provided.
- The purchase price cannot be less than 95% of the sales price.
- The purchase price cannot exceed the MSRP.
- The purchase types are:

  - Bank Finance
  - Cash
  - Dealer Finance
- Upon saving purchase information in the database the vehicle should no longer show up in inventory searches and should only be viewed on the sales report.
- The purchase date should be saved (today).
- The sales person should be saved (currently logged in user).

Admin/Vehicles

This page allows admin to edit inventory.

- It should only be accessible to users in the "Admin" role.
- It should only show inventory that is available for purchase.
- The search should utilize a jQuery AJAX call.

Admin/AddVehicle

This page allows adding vehicles to inventory.

- Only "Admin" role users may access this page.
- Make should be populated from the database.
- When a make is selected, the model combo box should filter to only show models associated with the selected make.
  a. Use a jQuery AJAX call to retrieve models associated with a given make.
- The type will be either New or Used.
- The body styles are Car, SUV, Truck, and Van.
- Year must be a 4 digit year between 2000 and the current year + 1
- Transmission should be Automatic or Manual
- Color and Interior colors should be a standard list of colors found in automobiles. For this assignment simply provide at least 5 colors to choose from.
- If the type is new, mileage should be between 0 and 1000. Used vehicles must be 1000+.

- VIN # is required. For extra credit research the VIN # format and enforce it. (optional)
- MSRP and Sale Price must be a positive number. Sale Price must not be greater than MSRP.
- A description is required.
- A picture is required.
    - Picture uploads must be .png, jpg, or jpeg format.
    - The uploaded file name should be ignored. Files should be named ‘inventory-x’ where x is the database ID of the vehicle.
    - Images should be stored in a folder called images/ in the UI project.
- Upon saving the page should be redirected to Admin/EditVehicle

Admin/EditVehicle

This page allows editing vehicles in inventory.

- Only “Admin” role users may access this page.
- The same rules as Admin/AddVehicle apply to this page with some exceptions.
    - A picture is optional. If a picture is uploaded, it should replace the existing picture in the images/ folder.
- The “feature this vehicle” box, when checked, should show this vehicle on the Home/Index page.
- The delete button, when pressed, should prompt the user whether they are sure before deleting.
- Deleting a vehicle should also delete the image file from the images/ folder.

Admin/Users

This page allows viewing application users.

- Only “Admin” role users may access this page.
- Users may not self-register, so no registration page should be provided. Admin users add all other users to the system manually.

Admin/AddUser

Allows for adding a new user.

- Only “Admin” role users may access this page.
- All fields on this form are required.
- The roles are Disabled, Sales, and Admin.

Admin/EditUser

Allows for editing a user.

- Only “Admin” role users may access this page.
- The password and confirm password fields will change a user’s password if filled in, otherwise the password should remain the same.
- If a user is changed to the “Disabled” role their login information should be rejected when they attempt to login to the site.
  - You may do this by writing custom code in the Login action of the Account controller or by storing user details in a separate table from the AspNetUsers table and deleting the AspNetUsers record when disabled is selected.
  - Regardless of which method you choose, the sales report should still show data for disabled users.

Account/ChangePassword

Allows a user to change their password.

- Only “Admin” and “Sales” roles may access this page.

Admin/Makes

Allows an admin to add a new make.

- Only “Admin” roles may access this page.
- Makes may not be deleted, only added.

Admin/Models

Allows an admin to add a new model.

- Only “Admin” roles may access this page.
- Models may not be deleted, only added.

Admin/Specials

Allows an admin to add and remove specials

- Only “Admin” roles may access this page.
- Deleting a special should prompt for confirmation before deleting.
- Title and Description are required.

Reports/Index

Provides a list of system reports.

- Only “Admin” roles may access this page.

Reports/Sales

Provides an aggregate of total sales and count of vehicles sold for sales staff.

- Only “Admin” roles may access this page.
- Search should be a jQuery AJAX call.
- If no filters are provided, sales for all time should be returned.
- If filters are provided the query should filter accordingly by user and/or date range.
- Omitting the "from date" means “sales for all time up to the to date”.
- Omitting the “to date” means “all sales after the from date”

Reports/Inventory

Provides an aggregate of new and used vehicles in stock and their stock value.

- Only “Admin” roles may access this page.
- Data should be grouped by year, make, and model.
- Stock value is the sum of the MSRP of the vehicles.

Account/Login

You may use the standard login provided by ASP.NET Identity. This page should not be linked in the navigation bar as we do not want the public to see it. This page and the change password page will be navigated to directly by our staff who will bookmark the links.

Submitting your Assessment

When you are satisfied that all of the objectives are completed take the following actions:

1. Submit a Crucible ticket for a code review.
2. Submit the link to your Crucible code review request.
3. A staff member will schedule a time to review your code with you. Be prepared to answer questions about your code and thought processes.

Mastery Car Dealership Rubric

Criteria	Ratings			Pts
☐ Mastery - Specifications Completed Meets Expectations All requirements in the specification assigned are completed and work according to specification. The submitted application: Utilizes a layered architecture with a UI, BLL, Data, and Test project. Contains classes that demonstrate a reasonable level of separation by responsibilities. Utilizes dependency injection between the BLL and Data tiers. Uses appropriate error handling and user input validation. Properly secures pages and database objects using ASP .NET Identity and SQL security features. Needs Improvement Requirements are not met to specification such as: Application code is not in a working state. Unsatisfactory or incorrect separation of concerns by layer. Classes frequently contain multiple, disparate responsibilities. Dependency injection is not used or used incorrectly. Inadequate unit test coverage. Inadequate input validation and/or error handling. Inadequate security utilization. threshold: 60.0 pts	**60 pts** **Meets Expectations**	**30 pts** **Needs Improvement**	**0 pts** **No Submission**	60 pts

Criteria	Ratings			Pts
Mastery - Apprentice Demonstrates Understanding of Code Meets Expectations Apprentice demonstrates an understanding of creating full-stack web applications. This includes: Cient side programming using HTML, CSS, and JavaScript. Server side programming with ASP.NET MVC and Web API. How HTTP Requests and Responses are handled. How data binding works. How routing works. The interplay between Models, Views, and Controllers. Validation in the UI layer: jQuery Validate Data Annotations IValidateableObject Securing Applications ASP .NET Identity SQL Server security Role based security Apprentice can describe the mechanism by which medium-sized projects are planned and organized with adherence to good design principles such as layering, dependency injection, and unit testing. Additionally the apprentice can describe what types of features are best implemented client side, server side, or in the database. Needs Improvement Apprentice poorly describes or demonstrates a lack of understanding of creating full-stack web applications. threshold: 10.0 pts	**20 pts Meets Expectations**	**10 pts Needs Improvement**	**0 pts No Submission**	20 pts

Criteria	Ratings			Pts
☐ Code Style Meets Expectations All code uses professional naming conventions, indentation, etc. Needs Improvement Some code does not meet professional naming conventions, indentation, etc. threshold: 10.0 pts	**10 pts** **Meets Expectations**	**5 pts** **Needs Improvement**	**0 pts** **No Submission**	10 pts
☐ Code Review Submitted to Crucible Meets Expectations Crucible submission was done properly and able to be reviewed by SG Staff. Needs Improvement Crucible submission is missing some information or linked improperly. threshold: 5.0 pts	**5 pts** **Meets Expectations**	**3 pts** **Needs Improvement**	**0 pts** **No Submission**	5 pts
☐ Submit Engineering Notebook threshold: 5.0 pts	**5 pts** **Meets Expectations**	**3 pts** **Needs Improvement**	**0 pts** **No Submission**	5 pts
				Total Points: 100