HTTP5101 Assignment 3

By: Christine Bittle

This assignment requires you to build a minimum viable product (MVP). The MVP is to build ‘read’ functionality on the provided teachers mysql table using WebAPI and MVC architecture pattern. This will include:

Type	Description	File
-	A class which connects to your MySQL database	/Models/SchoolDbContext.cs
Controller	A WebAPI Controller which allows you to access information about teachers	/Controllers/TeacherDataController.cs
Controller	A Controller which allows you to route to dynamic pages	/Controllers/TeacherController.cs
Model	A Model which allows you to represent information about a teacher	/Models/Teacher.cs
View	A View which uses server rendering to display teachers from the MySQL Database	/Views/Teacher/List.cshtml
View	A View which uses server rendering to display a teacher from the MySQL Database	/Views/Teacher/Show.cshtml

Bonus!
Earn Initiative marks by improving upon the MVP. Some suggestions:

- Build a search interface to find a teacher (by name, hiredate, salary)
- Display the courses taught by a teacher in /Views/Teacher/Show.cshtml
- Build the MVP for read functionality on the Students table
- Build the MVP for read functionality on the Classes table
- Update Views/Shared/_Layout.cshtml and /Content/Site.css to significantly change the layout and design of your web application.

5101 Assignment Rubric

	Level 1 (0-25%)	Level 2 (25-50%)	Level 3 (50-75%)	Level 4 (75-100%)
Quantitative	Multiple (4+) Quantitative issues. The project needs significant improvement to meet professional development standards.	Several (2+) Quantitative issues.There are several areas of improvement needed to meet professional development standards.	One Quantitative issue. A few fixes can bring this to professional quality standards.	Zero Quantitative issues. Work is at a professional level. Work is complete, maintainable, scalable, robust, efficient, extensible, and reusable.
Qualitative	Multiple (4+) Qualitative issues. The project needs significant improvement to meet professional development standards.	Several (2+) Qualitative issues. There are several areas of improvement needed to meet professional development standards.	One Qualitative issue. A few fixes can bring this to professional quality standards.	Zero qualitative issues. Work is at a professional level. Work is concise, readable, well-documented, tested, and includes evidence of debugging.
Semantic	Multiple (4+) Semantic issues. The project needs significant improvement to meet professional development standards.	Several (2+) Semantic issues. There are several areas of improvement needed to meet professional development standards.	One Semantic issue. A few fixes can bring this to professional quality standards.	Zero semantic issues. Work is at a professional level. The work achieved is considered and aligned with the context of the project.
Initiative	The content of the work meets the bare minimum requirements.	The content of the work meets the requirements, and there is an attempt to try something new. Not working code is commented out.	The content of the work exceeds the expectations of the assignment. Not working code is commented out.	The content of the work exceeds the expectations of the assignment, and the code runs adequately.

“Perfection is not attainable, but if we chase perfection we can catch excellence.”

- Vince Lombardi

In our classes we discuss the ideas of *quantitative, qualitative, and semantic* concerns with our work, with several examples of improvements that we can make. The chart below aggregates some quality coding practices into these overall categories. No codebase is perfect, however, *“professional quality”* work is considered to show acknowledgement and attention to these dimensions.

Category	Focus	In-depth
Quantitative	**Completeness (a.k.a MVP)** : Does your work achieve the required task?	“What does MVP mean and why you need It” By Mariia Lozhka
	Maintainability: Is it easy to adjust your code if you have to?	“For Secure Code, Maintainability Matters” by G. Ann Campbell
	Scalability : Will your codebase easily grow to meet new requirements? Will it work well with a larger input size?	“What the hell is scalable code anyway?” by Saras Arya
	Robustness : Will your code still work given unexpected circumstances?	“How to write robust code” by Salvatore Iovene
	Efficiency : Does your solution avoid detours?	“Big-O Notation Explained with Examples” by Vineet Choudhary
	Reusability: Can parts of this work be reused for similar problems?	“The Challenge of Code Reuse” by Richard Bellairs
Qualitative	**Concise**: Does your code avoid redundancies?	“Don’t repeat yourself” - Wikipedia
	Readability : Is it easy to read your code at a glance and understand how it works?	“What is Code Readability?” by Aakansha Damani
	Documentation : Do you explain how your code works?	“The eight rules of good documentation” by Adam Scott
	Testing : Are you testing your code for different circumstances?	“The A-Z Guide to the Software Testing Process” By Ulf Eriksson

	Debugging : Are you using debugging techniques?	“10 Debugging Tips for Beginners” By Hartley Brody
Semantic	**UI/UX Considerations** : How will a user interact with your product?	“Part 1 — The Design of Everyday Things (Revised & Expanded Edition)—Book Summary & Key Points” Book by Donald Norman, Article by Lim Zhiyang
	Data Structure Considerations : Is the data represented well?	“Database Design and Modeling Fundamentals” By Brent Huscher
	Convention : Is the project structure similar to other projects on the same framework?	"Convention over configuration” - Wikipedia