Database

* The Properties table has several nvarchar(max) columns which can’t be indexed so database performance problems are imminent when table size will be getting larger and larger
* The Properties table has no indexes even website allows to search properties by street name and description. So, after fixing mentioned above issue with nvarchar(max) columns the search query can be optimized by creating index on StreetName and Description columns.
* Store date and time in UTC format and write code to convert it back to local time based on timezone difference between UTC and local time

Code

# What is good about it?

* Code is very well structured and easy to follow and understand
* Great separation concern between different logical entities such as view model builders and command handlers
* Each view model builder and command handler follows SRP (single responsibility principle) which makes it very easy to test and maintain
* Consistent naming convention throughout the entire code-base
* Good unit testing coverage
* Good project organization so all related items are located next to each other in the folder hierarchy

# What is bad about it?

* All 3 tiers such as presentation, business logic and data tiers are merged within one web application. So, although code is well structured this architecture is not suitable for the large enterprise systems which typically contain more than one client interface (web, mobile, desktop, B2B integration api) which needs to consume the same dataset
* Need to do some refactoring to reuse existing code going forward in middleware services or in web api used by mobile or by single page application. For example, all commands need to be moved in the separate assembly
* Missing business logic layer results in view mode builders fetching data directly from the database and mapping database entities to the view models. Although It works well for this web app the additional service layer needs to be created in order to share business logic and data between different applications and services. Each application consuming the same dataset needs to share the same services api to maximize code reuse and simplify maintaince but have own version of the view model. Each application has different view model requirements whether it is SPA, Multi-page or mobile so sharing it between them will results in overexposing or underexposing data
* Had to add missing jqueryval bundle to the layout page in order enable validations on the client side
* Missing ValidateAntiForgeryTokenAttribute attribute on all action methods that indented to be protected against cross-site request forgery such as login, resetting password, creating property and etc.
* Avoid using string literals if literal string used in more than one place in the code such as property type for example
* Missing default error page and global exception handler for handling all unhandled errors and logging exception error messages in the log file