WPF Interview Assignment

# Introduction

This assignment is simple but not trivial and would require up to 8 hours of work depending on WPF expertise level. Assignment results in a functioning WPF application implementing simple TODO list. Assignment is based on the well-known <http://todomvc.com/> benchmark for selecting JavaScript frameworks. Thus, you can look at existing implementations to see how application should behave and use any graphics assets (icons, backgrounds, etc.).

# Application Functionality

#mark all

#main

#footer

# Todo app screenshotNo todos

When there are no todos, #main and #footer should be hidden (see another picture below).

## New todo

New todos are entered in the input at the top of the app (see “*What needs to be done?”* prompt in the picture above). The input element should be focused when the application is loaded preferably using (think if XAML only is enough). Prompt is replaced with the first typed letter of the todo. Pressing Enter creates the todo, appends it to the todo list and clears the input. Make sure to trim the input and then check that it's not empty before creating a new todo.

## Mark all as complete

This checkbox toggles all the todos to the same state as itself. Make sure to clear the checked state after the "Clear completed" button is clicked. The "Mark all as complete" checkbox should also be updated when single todo items are checked/unchecked. E.g. when all the todos are checked it should also get checked and when at least one is not checked among all existing todos it gets unchecked. This checkbox is hidden when there are no todo items.

## Item

A todo item has three possible interactions:

1. Clicking the checkbox marks the todo as complete by updating its completed value.
2. Double-clicking the rest of the item activates editing mode
3. Hovering over the todo shows the remove button (at the right side of the item)

## Editing

When editing mode is activated it will hide the other controls (checkbox, remove button) and activate an input that contains the todo title, which should be focused. The edit should be saved on both focus out and enter, and the editing should be deactivated. Make sure to trim the input and then check that it's not empty. If it's empty the todo item should instead be removed. If escape is pressed during the edit, the edit state should be deactivated and any changes be discarded.

## Counter

Displays the number of active todos (yet to be completed) in a pluralized form. Make sure the number is displayed in bold. Also make sure to pluralize the item word correctly: 0 items, 1 item, 2 items, etc. Example: **2** items left.

## Clear completed button

Displays the number of completed todos, and when clicked, removes them. Should be hidden when there are no completed todos.

## Filtering

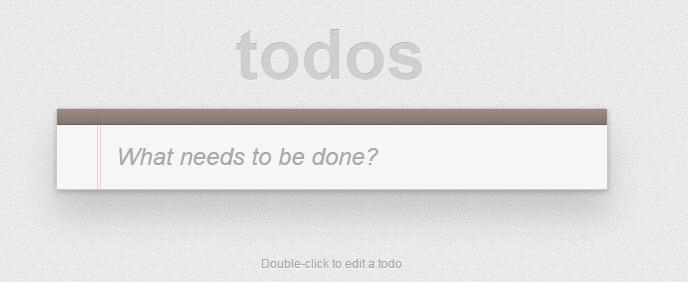
The following filters should be implemented: All, Active, Completed. When the filter changes the todo list should be filtered on a model level and filter should be displayed in bold. When an item is updated while in a filtered state, it should be updated accordingly. E.g. if the filter is Active and the item is checked (completed), it should be hidden. Make sure the active filter is implemented in such way that it can be persisted if necessary.

## Persistence

No persistence is necessary, all structures are in memory.

# Assignment

Implement TODOS application as a single window with content centered vertically (can be fixed size) and **todos** label at the top.



Implementation language is C# and UI technology is WPF. Please, provide zipped Visual Studio solution with complete source code for assignment review.

Apply MVVM pattern to structure the solution. You can use basic libraries of your choice (for example, MVVM light or WAF). Implement custom control to support prompt (watermark).

There is no need to optimize LOC or eliminate repetitive code. It is enough to indicate in comment that the particular code can be abstracted or improved in real application. Provide enough indication in what way it can be done and be ready to defend.

WPF visual styling should be done in separate resource file.

Though persistence is not required, application design should allow implementing persistence without major changes to its structure. Persistence can be understood as closing and then re-opening application restores its state exactly, except for window position/size, focus, scrolling and editing states.