MVC TASK 3

[Document subtitle]

LinkedIn Article:

https://www.linkedin.com/pulse/understanding-filters-model-binding-aspnet-core-mvc-mohamed-afify-ecy5f/

3 Cases When a ViewModel Is a Must

1-Security Case

- **Problem:** If you pass your Entity model directly to the View, it may contain sensitive fields (password hashes, admin flags, hidden IDs…).
- **Solution:** Use a ViewModel containing only the properties you actually need to display or post back.
- **Why:** It prevents over-posting and stops users from tampering with fields they should not see or edit.

Example: A User entity contains PasswordHash or IsAdmin. Using a ViewModel ensures only non-sensitive fields go to the View.

2-Combining Multiple Models in One View

- **Problem:** A page often needs data from multiple tables/entities at once (like Department + Students + Courses).
- **Solution:** Build a ViewModel that aggregates the needed properties from all those sources.
- **Why:** This keeps the View strongly typed and clean while avoiding multiple separate models.

Example: A "Department Details" page shows department info, a list of students, and available courses simultaneously.

3-Data Shaping / Formatting for the UI

- **Problem:** The database fields aren't always in the format you want to display or capture from the user (e.g., dates, currencies, combined text).
- **Solution:** A ViewModel lets you reshape or re-type those fields for the View.
- **Why:** This separates the persistence model from the presentation model, making your UI simpler and clearer.

Example: A DateTime field in the database but you want a formatted string in the UI.