Question Answering Assignment

1. State 3 cases that ViewModel is a must and one of these cases Security case.

Three cases where using a ViewModel is a must are:

1. **Security Case**
   When exposing data to the view, a ViewModel is essential to avoid unintentionally exposing sensitive data from domain or entity models. For example, if the domain model contains fields like hashed passwords or sensitive user information, using a ViewModel ensuring only the data needed for the view is included prevents security risks. The ViewModel acts as a Data Transfer Object (DTO) that filters out sensitive or unnecessary properties, protecting the data exposed to the client or front-end. Additionally, any input received through a ViewModel should be validated and not trusted directly because users can manipulate it.

2. **Tailoring Data for a View**
   When the data required by a particular UI view is a subset or combination of multiple models or needs to be structured differently from the database entities, a ViewModel is necessary. It simplifies the binding by shaping the data specifically for the UI, improving separation of concerns and maintaining clarity.

3. **Complex Input Forms or UI State**
   For views requiring input forms with additional UI state or metadata (e.g., validation messages, dropdown options, checkboxes states), the ViewModel can encapsulate all relevant UI-related data separately from the domain models, making it easier to manage user input and state within the UI lifecycle.

Security Focus with ViewModel Explained

- Never put sensitive data directly into the ViewModel since it is sent to the client browser and can be inspected or manipulated.
- Always strip down to only the properties the page requires.

- Treat ViewModel as an input/output of a REST API: validate and re-calculate critical data on the server side.
- Use techniques such as signing or encrypting sensitive parts of the ViewModel to protect against tampering.
- Enforce authorization and permission checks in the business layer based on ViewModel data to ensure security.

These practices ensure that ViewModels protect the system's security by limiting attack surfaces and controlling data flow