Part 1- Team Collaboration

Task 1- How can you improve the code and refactor it?

- **Separation of Concerns -** *'someVC'* class is doing a lot, including networking (fetchData), UI animations (showDetail and closeshowDetails), and collection view delegate methods. Consider separating concerns into different classes or extensions for better maintainability.
    - Create a dedicated networking class to handle network-related functionality.
    - Create a ViewModel for '*someVC'* that uses Networking class to fetchdata. ViewModel helps in separating the business logic from the UI components, making your code more testable and maintainable
    - Create separate extension of '*someVC'* for collection view delegate methods.
- **Network Request Handling** - Error handling for the network request is missing. Handle errors appropriately, and consider using Result types for a cleaner approach.
- **Reusability and Constants** - Use constants or enums for reuse identifiers and magic numbers to make the code more readable and maintainable. Consider creating a dedicated class for networking and reusable constants.
- **View Lifecycle** - Perform tasks related to setting up the view hierarchy and initial data loading in the *viewDidLoad* method rather than *viewWillAppear*.
- **UI updates** - Perform UI updates on the main thread after receiving data from a network request.
- **Asynchronous Code** - The `data` parameter in the completion handler of the network request is optional. Safely unwrap it before using it to avoid runtime crashes
- **CollectionViewCell -** custom cell "someCell" is registered in the *viewWillAppear* method, but it's not being used in the *cellForItemAt* method. It can be solved with following steps
    - Move the cell registration code to a more appropriate place, such as *viewDidLoad*
    - In the *cellForItemAt* method, dequeue the cell using the registered reuse identifier.
- **Code Duplication:** Both showDetail and closeshowDetails methods share similar code for updating constraint values and adding/removing subviews.
    - Create a common private method to handle the shared functionality, this method should take parameters or have a clear logic to determine whether to show or close the detail view.
    - Replace the code in both methods with calls to the common method, specifying the appropriate parameter.
    - **Assumption**: Not sure about the exact UI design but right now UI is updated by two modifications updating constraint value and adding/removing subview , I think one of these should be used not both
- **Code Formating** - Maintain consistent code formatting..
    - Ensure indentation/spacings are consistent, and code is well-formatted for readability
    - Change Method name to "*closeshowDetails*" to "*closeDetails*"

- **UIDevice Extension** - Instead of creating a global method for iPhone creating an extension for UIDevice is a good approach to encapsulate the logic related to checking whether the device is an iPhone. This makes the code more modular and adheres to the principles of encapsulation and separation of concerns.

Task 2- Write a feedback for the author of this PR?

Thank you for your contribution to the project with your recent pull request. I've carefully reviewed the changes, and overall, it's evident that you've put effort into achieving the desired functionality. I appreciate your dedication to enhancing the codebase.

Here are some observations and suggestions:

- **Separation of Concerns -** Consider separating concerns within the `someVC` class. Grouping related functionality into extensions or separate classes can enhance maintainability.
- **Network Request Handling-** Implement proper error handling in the network request (`fetchData`) for a more robust solution**.**
- **Reusability and Constants -** Use constants or enums for reuse identifiers and magic numbers to improve code readability.
- **View Lifecycle -** Consider moving the data fetching logic from `viewWillAppear` to `viewDidLoad` if it's not necessary to fetch data every time the view appears.
- **UI updates** - Perform UI updates on the main thread after receiving data from a network request.
- **Asynchronous Code** - Safely unwrap the optional `data` parameter in the network request completion handler to prevent runtime crashes.
- **UIDevice Extension** - Create an extension for `UIDevice` to encapsulate the logic for checking if the device is an iPhone.
- **Code Duplication** - The `showDetail` and `closeshowDetails` methods share common logic. Refactor this into a private method to avoid duplication and improve maintainability.
- **Code Formatting** - Ensure consistent code formatting and indentation for better readability.

Your efforts are commendable, and these suggestions are meant to enhance code quality and maintainability. Remember, these are recommendations, and you can prioritize based on the project's needs.
Thank you for your contribution, and I look forward to seeing the improvements. If you have any questions or need further clarification, feel free to reach out.