1. Naming Conventions

   Avoid short forms like VC use fullnames eg: detailVC, someVC

   Use camel case for ll variables and methods in your code. Eg: closeshowDetails

2. IBOutlets should be weak to avoid retain cycle

```
 @IBOutlet var detailViewWidthConstraint: NSLayoutConstraint!
   @IBOutlet var collectionView: UICollectionView!
   @IBOutlet var detailView: UIView!
```

3. Remove Unused properties:

   remove collectionViewTopConstraint, consider removing it if its unused.

```
@IBOutlet weak var collectionViewTopConstraint: NSLayoutConstraint!
```

4. Call super.ViewWillAppear to preserve expected view  behaviours

```
override func viewWillAppear(_ animated: Bool) {
      fetchData()
```

5. Hardcoded URL:

   Avoid using hardcoded URL. Add as constant or make it dynamic.

```
let url = URL(string: "testreq")!
```

6. Adding / Removing detail VC doesn't make sense please correct it.

```
closeshowDetails ()  => self.detailVC?.removeFromParent()
showDetail() =>  self.view.addSubview(self.detailView)
```

7. Implement Data response properly

   Below code always fails. You may need to use JSONDecoder to decode data here.

```
self.dataArray = data as? [Any]
```

8. Error Handling:

   add proper error handling to check for errors in fetchData()  and handle them.

```
let task = URLSession.shared.dataTask(with: url) { [self](data, response, error) in
          self.dataArray = data as? [Any]
          self.collectionView.reloadData()
      }
```

9. Use Constant Struct:

   Create Constant structs for animation durations, detail view width. Etc . This will increase maintainability

   eg:

```
showDetai()l,

func collectionView(_ collectionView: UICollectionView,
layout collectionViewLayout: UICollectionViewLayout,
        sizeForItemAt indexPath: IndexPath) -> CGSize ,

        func collectionView(_ collectionView: UICollectionView, layout
      collectionViewLayout: UICollectionViewLayout,
      minimumLineSpacingForSectionAt section: Int),
```

10. CollectionView Datasource Implementation:

    Please add cell implementation in CollectionViewDataSource methods

```
func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
  UICollectionViewCell {
    return UICollectionViewCell()
  }
```

11. Reference Cycle

    There could be a possible reference cycle in fetchData as you are strongly referring self inside async block. Please use [weak self]

```
let task = URLSession.shared.dataTask(with: url) { [self](data, response, error) in
        self.dataArray = data as? [Any]
        self.collectionView.reloadData()
    }
```

12. Group methods better and organize the code

    Use extensions for implementing delegates. Group same behaviour like methods together. Add comments.

13. Follow SOLID Principles

    Looks like the viewcontroller is responsible for fetchingData, setting up collection view, populatiing cell etc. Please divide the work to follow SRP better.