Software Engineer (iOS)

Take-Home Exercise

At Diligent, we use a *Structured Interview* process to hire in a more predictable and inclusive fashion. By focusing the interview process on a realistic task, we can better evaluate a candidate's suitability, while allowing all types of individuals (irrespective of their communication style or pattern) to demonstrate their ability.

For the Software Engineer interview, there are two top-level parts to the role we want to assess in this task:

- Team Collaboration
- Technical Ability

The document is split up as such. If anything is not clear, please make reasonable assumptions (and indicate them accordingly) and continue. You are welcome to write as much as you want, but please keep in mind that clarity and conciseness are critical in business communication; so more is not always better. Please be prepared to discuss your answers during your technical interview.

We thank you in advance for your time and willingness to participate in this Structured Interview process!

How to Submit answers?

- *Create a GitHub Repository, with the following Name (YourName_DILTest)*
- *Create 2 Folders: Part 1 & Part 2.*
- *Upload your files with answers & project to dedicated folders.*
- *Submit your Repository link.*

Part 1

Team Collaboration

Our R&D teams are like soccer teams (we all know it is “football”, but play along with me): we’ll only win if we play as *one unit* and pass the ball appropriately. We create team players who will learn from each other, help each other grow, pass the ball when required, and ask for help. Team collaboration is thus essential to our culture and our success.

Collaboration opportunities are prevalent, but for this interview task assignment, we’ll focus on the very typical scenario of Pull Requests.

Tasks:

1. How can you improve the code and refactor it?
2. Write a feedback for the author of this PR.

```
import UIKit

func isIPhone() -> Bool {
    return UIDevice.current.userInterfaceIdiom == .phone
}

class someVC:
UIViewController,UICollectionViewDelegate,UICollectionViewDelegateFlowLayout,UICollectionViewDataSource {

    @IBOutlet weak var collectionViewTopConstraint: NSLayoutConstraint!
    @IBOutlet var detailViewWidthConstraint: NSLayoutConstraint!
    @IBOutlet var collectionView: UICollectionView!
    @IBOutlet var detailView: UIView!

    var detailVC : UIViewController?
    var dataArray : [Any]?

    override func viewWillAppear(_ animated: Bool) {

        fetchData()

        self.collectionView.dataSource = self
        self.collectionView.delegate = self
        self.collectionView.register(UINib(nibName: "someCell",
                                           bundle: nil), forCellWithReuseIdentifier: "someCell")
        self.navigationItem.rightBarButtonItem =  UIBarButtonItem.init(title:NSLocalizedString("Done", comment:
""), style: .plain, target: self, action: #selector(dissmissController))

    }

    @objc func dissmissController () {}

    func fetchData() {

        let url = URL(string: “testreq")!
         let task = URLSession.shared.dataTask(with: url) { [self](data, response, error) in

            self.dataArray = data as? [Any]
            self.collectionView.reloadData()
        }

        task.resume()
    }


@IBAction func closeshowDetails () {
        self.detailViewWidthConstraint.constant = 0

        UIView.animate(withDuration: 0.5, animations:
                        {

            self.view.layoutIfNeeded()

        })
        { (completed) in
            self.detailVC?.removeFromParent()
        }
    }

    @IBAction func showDetail () {

        self.detailViewWidthConstraint.constant = 100

        UIView.animate(withDuration: 0.5, animations:
                        {

            self.view.layoutIfNeeded()

        })
        { (completed) in
            self.view.addSubview(self.detailView)
        }
    }

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

        var widthMultiplier: CGFloat = 0.2929
        if isIPhone() {
            widthMultiplier = 0.9
        }
        return CGSize(width: view.frame.width * widthMultiplier ,
                      height:  150.0)
    }

    func collectionView(_ collectionView: UICollectionView, layout
                        collectionViewLayout: UICollectionViewLayout,
                        minimumLineSpacingForSectionAt section: Int) -> CGFloat {

        let frameWidth = (view.frame.width * 0.2929 * 3) + 84
        var minSpacing: CGFloat = (view.frame.width - frameWidth)/2
        if isIPhone() {
            minSpacing = 24
        }
        return minSpacing
    }
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return  self.dataArray?.count ?? 0
    }
    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) ->
UICollectionViewCell {
        return UICollectionViewCell()
    }
    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        self.showDetail()
    }
}
```

Part 2

Technical Ability

Create a ghost drawing board app, where you can select among three different colors (red, green, blue) to draw with and then you can erase the drawing with an eraser selection. That's pretty standard drawing board but what about the ghost part? Well, here is how it plays:

- Anything which you are drawing should appear on screen after certain seconds. For the case of :
    - Red 1 second.
    - Blue 3 seconds.
    - Green 5 seconds.
    - Eraser works on with the 2 seconds delay too.
- The delay applies to anytime whenever you are done drawing with any of the color. For example If you picked red and started drawing, there wouldn't be any drawing on the screen unless you would end sliding on the screen and lift the finger. After 1 second, it will start drawing on the screen at same exact pattern as you slide your finger.