SINGAPORE UNIVERSITY OF SOCIAL SCIENCES

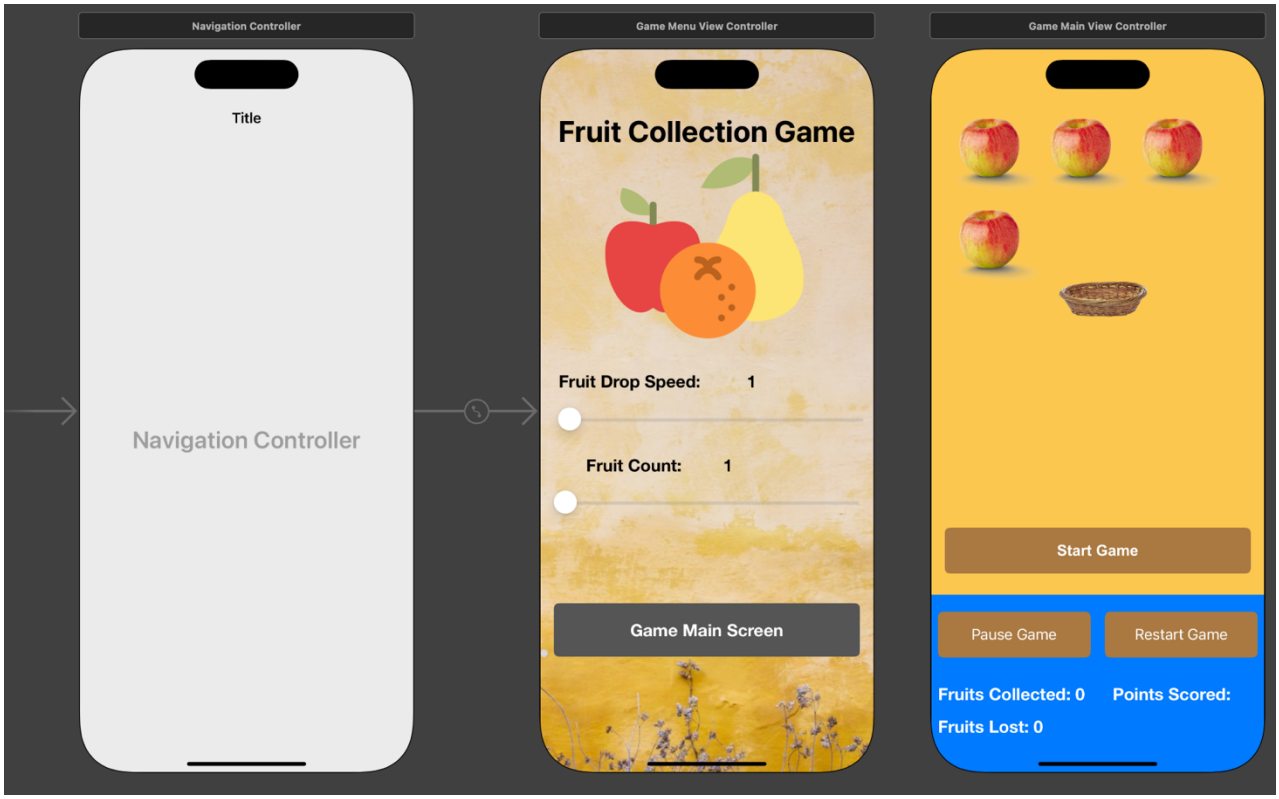
MTD367
iOS App Development
July 2024
TMA02

Name:	Yang Xian Wei Shawn
T-Group	T01
Date Submitted	20 September 2024

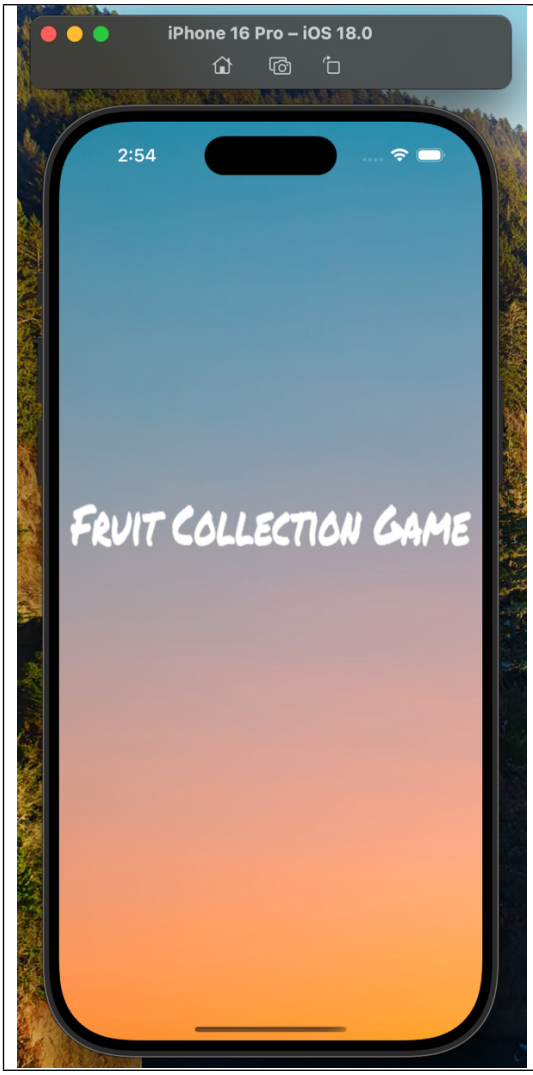
Question 1 (40 marks)

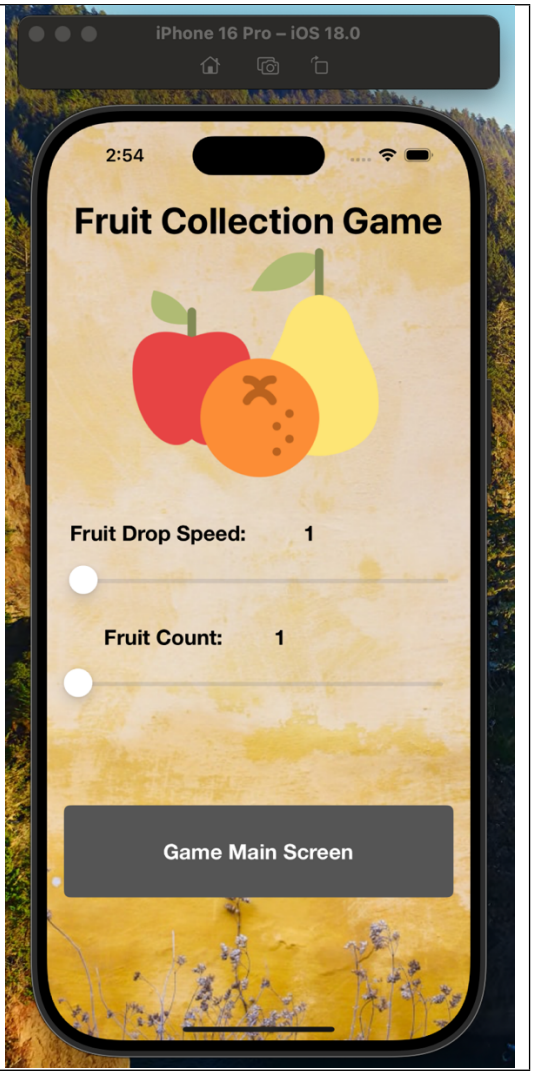
Question 1(a) (10 marks)

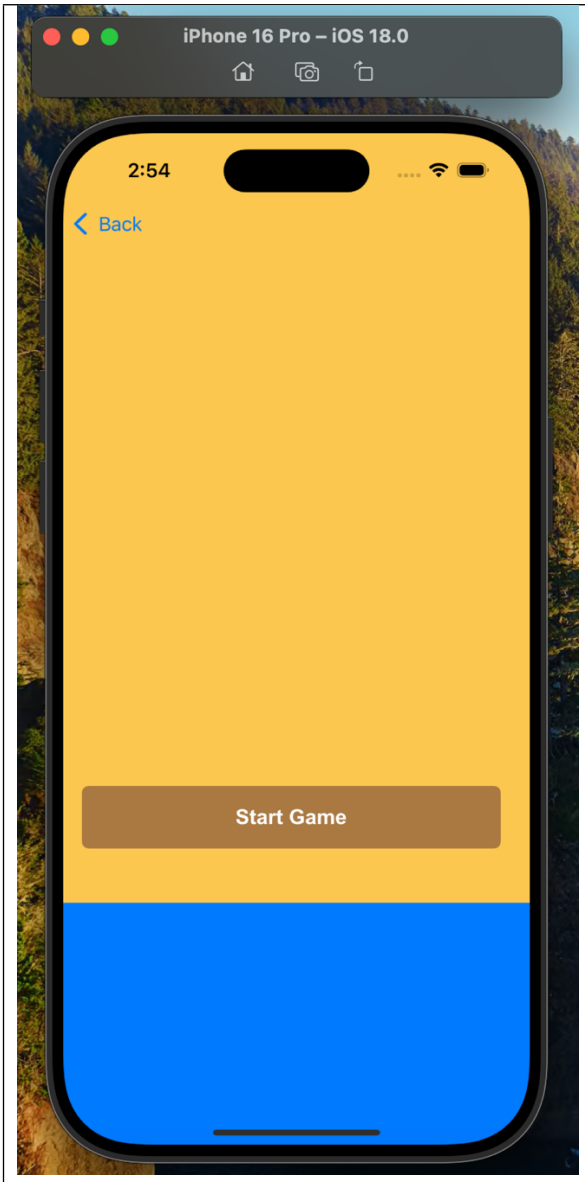
App Workflow:

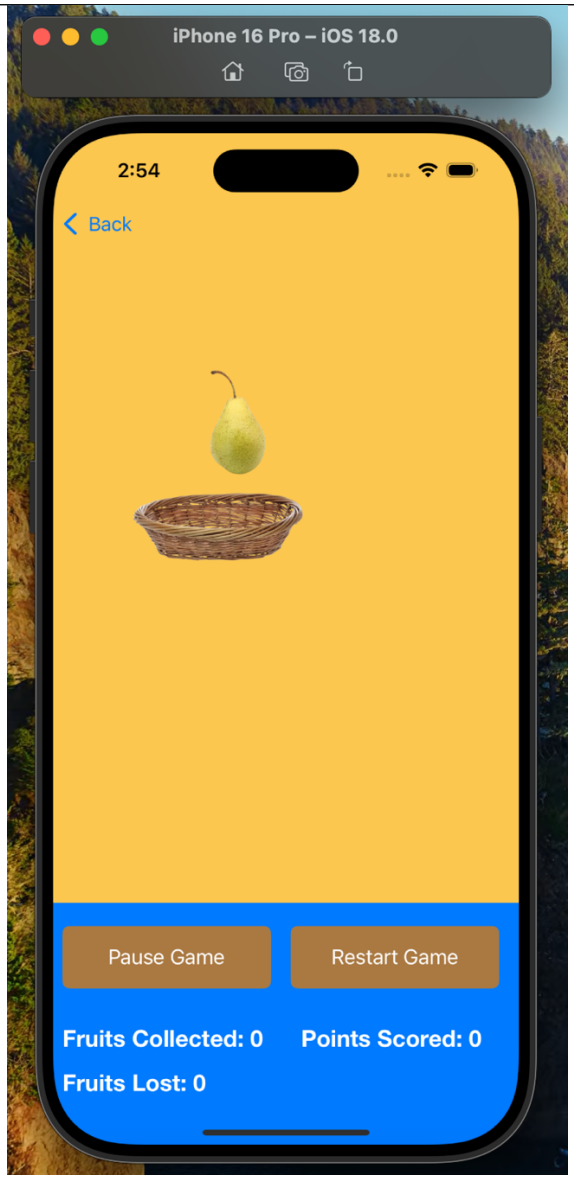


Simulator Flow:


iPhone 16 Pro – iOS 18.0
2:54
Fruit Collection Game


iPhone 16 Pro – iOS 18.0
2:54
Fruit Collection Game
Fruit Drop Speed: 1
Fruit Count: 1
Game Main Screen





GameMenuViewController UIViews used and their related functions:

- I used several UILabel's to display the title of the game called "Fruit Collection Game", a title for each UISlider such as "Fruit Drop Speed" and "Fruit Count" as well as 2 labels to display the number value for each slider.

- I used the UIImageView to display several UIImageView's such as a logo for the game and a background image instead of a plain background color to make the game app look more dynamic.

GameMenuViewController UIControls used and their related functions:

- I used a control called UIButton titled "Game Main Screen" that when selected will direct users to the next view called "GameMainViewController" where users can play the game.

- I used a control called UISlider and defined 2 UISlider's, 1 slider to determine the drop speed for the fruit that falls down the screen in the "GameMainViewController" where the value 1 being the slowest speed and the value 4 represents the maximum and fastest speed that which a fruit can fall down the screen at.

  The 2nd slider is used to determine the number of fruits that can fall off the screen in the "GameMainViewController" where the value 1 is the default value which means that only 1 fruit by default can fall off the screen and the maximum value is 4 which means that there can be a maximum of 4 fruits that can fall off the screen simultaneously.

GameMainViewController UIViews used and their related functions:

- I used the UIImageView to display several UIImageView's such as 4 fruit images which would be the objects that will fall off the screen and a basket image to represent the container that would be used to collect the fruits.

- I used 3 UILabel's which are "Fruits Collected: 0", "Fruits Lost: 0" and "Points Scored:". The "Fruits Collected: 0" label is used to show the number of fruits that hit the container image, the "Fruits Lost: 0" label is used to show the number of fruits that do not hit the container image and instead fall off the screen and lastly the "Points Scored" label is used to show the number of points scored by the user when the a fruit hits the container image.

  Different types of fruits that hit the container image would allocate the user with a different number of points.

- I also used the UIView to create a rectangular UIView object with a system blue background to manage the 3 labels and the "Pause Game" and "Restart Game" buttons on the screen.

GameMainViewController UIControls used and their related functions:

- I used a control called UIButton and defined 3 UIButtons which are "Start Game", "Pause Game" and "Restart Game". The "Start Game" button is used to start the game that means that the fruit and basket images, buttons and labels would become visible and the fruit can start falling down the screen.

  The "Restart Game" button when double tapped would restart the game which means that when the container image position would be randomized and the 3 label values would be reset to 0.

  The "Pause Game" button is used to pause and resume the game. Users must first swipe left on the button to pause the game and swipe right on the button to resume the game. When users swipe left on the button to pause the game, the button text would be changed to "Resume Game" and when users swipe right on the button to resume the game, the button text would be changed to "Pause Game".

  When the game is paused, the fruit would stop falling down the screen and the basket image would not be allowed to be moved left and right. When the game resumes the fruit would be allowed to continue falling and the backet image would become interactable again.

App Flow in Sequence:

When a user launches the Fruit Collection Game, they will first encounter a launch screen that will be shown for a short period of time then the user would see the **GameMenuViewController** view where they can configure the fruit dropping speed and fruit count by moving the 2 sliders.

Once users are satisfied with their game configuration, they can tap the “Game Main Screen” button that will direct them to the **GameMainViewController** view where they will then tap the “Start Button” to launch the game.

Once the game is launched, a fruit will start falling down the screen, users can move the basket image left and right to collect the fruit and when a fruit is collected the “Fruits Collected:” value would increase by 1 and the number of points would also increase based on the type of fruit. An apple would provide 1 point, a pear would provide 2 points, a mango provide 3 points and a strawberry would provide 4 points.

When a fruit that is not caught by the basket and falls off the screen, the “Fruits Lost:” value will increase by 1. Users can also swipe left on the “Pause Game” button to pause the game and swipe right on the same button to resume the game. Users can also tap the “Restart Game” to restart the game where the “Points Scored:”, “Fruits Collected:” and “Fruits Lost:” values would be reset to 0 and the container image position would be randomised.

Question 1(b) (c) (d) (30 marks)

GameMenuViewController Source Code:

```
import UIKit

class GameMenuViewController: UIViewController {


    @IBOutlet weak var Text_Label: UILabel!


    @IBOutlet weak var Fruit_Img: UIImageView!


    @IBOutlet weak var StartGame: UIButton!


    @IBOutlet weak var fruit_drop_speed_slider: UISlider!


    @IBOutlet weak var slider_Label: UILabel!


    @IBOutlet weak var fruit_count_Slider: UISlider!


    @IBOutlet weak var fruit_count_Label: UILabel!
```

```
    override func viewDidLoad()
    {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
        fruit_drop_speed_slider.value = 0

        slider_Label.text = String(Int(fruit_drop_speed_slider.value))

        fruit_count_Slider.value = 0

        fruit_count_Label.text = String(Int(fruit_count_Slider.value))
    }

    @IBAction func btnTapped(_ sender: Any)
    {
        // MARK - Using Story board
        let storyboard = self.storyboard?.instantiateViewController(withIdentifier: "GameMainViewController") as! GameMainViewController

        // Pass the slider value to GameMainViewController
        storyboard.fruitDropSpeed = Int(fruit_drop_speed_slider.value)

        /*
         Pass the fruit count slider value to GameMainViewController
         Add 1 because slider starts at 0
        */
        storyboard.fruitCount = Int(fruit_count_Slider.value) + 1

        self.navigationController?.pushViewController(storyboard, animated: true)
    }

    @IBAction func sliderValueChanged(_ sender: UISlider)
```

```
    {
        slider_Label.text = String(Int(fruit_drop_speed_slider.value))
    }

    @IBAction func fruit_count_SliderValueChanged(_ sender: UISlider)
    {
        fruit_count_Label.text = String(Int(fruit_count_Slider.value))
    }
}
```

GameMainViewController Source Code:

```
import UIKit

class GameMainViewController: UIViewController {

    // IBOutlets for UI elements
    @IBOutlet weak var startBtn: UIButton!

    @IBOutlet weak var fruit_Img: UIImageView!

    @IBOutlet weak var container_Img: UIImageView!

    @IBOutlet weak var fruits_Collected_Count: UILabel!

    @IBOutlet weak var fruits_Lost_Count: UILabel!

    @IBOutlet weak var pause_resume_Btn: UIButton!

    @IBOutlet weak var restart_btn: UIButton!

    @IBOutlet weak var points_Count: UILabel!

    @IBOutlet weak var fruit_2: UIImageView!
```

```
@IBOutlet weak var fruit_3: UIImageView!

@IBOutlet weak var fruit_4: UIImageView!

// Game variables

// Random X position for the Fruits
var fruit_random_X = 0

// Y position for the Fruits
var fruit_position_Y = 0

// X position controlled by touch movement
var fruit_position_X = 0

// Number of times the player has collected the fruits
var fruits_Collected_Num = 0

// X position for the Container
var container_position_X = 0

// Number of times the player has lost the fruits (i.e., fruits fell off-screen)
var fruits_Lost_Num = 0

// Default Fruit Drop Speed
var fruit_drop_Speed: Int = 1

// Default fruit count
var fruit_Count: Int = 1

var points_Scored = 0

var is_game_Paused = false
```

```
// Add the flag
var is_fruit_Reset = false

var fruit_catching_GameTimer: Timer?

// To store the type of the currently falling fruit
var current_fruit_Type: UIImage?

let fruits: [UIImage] = [
    UIImage(named: "Apple")!,
    UIImage(named: "Pear")!,
    UIImage(named: "Mango")!,
    UIImage(named: "Strawberry")!
]

// Dictionary to map fruit types to point values
let fruit_Points: [UIImage: Int] = [
    UIImage(named: "Apple")!: 1,
    UIImage(named: "Pear")!: 2,
    UIImage(named: "Mango")!: 3,
    UIImage(named: "Strawberry")!: 4
]

override func viewDidLoad()
{
    super.viewDidLoad()

    // Do any additional setup after loading the view.

    /*
     Initial UI setup

     Show start button
```

```
*/
startBtn.isHidden = false

// Hide Container
container_Img.isHidden = true

fruit_Img.isHidden = true

// Hide fruits collected count label
fruits_Collected_Count.isHidden = true

// Hide fruits lost count label
fruits_Lost_Count.isHidden = true

// Hide pause and resume button initially
pause_resume_Btn.isHidden = true

// Hide restart button initially
restart_btn.isHidden = true

points_Count.isHidden = true

fruit_2.isHidden = true

fruit_3.isHidden = true

fruit_4.isHidden = true

// Initial position for the Fruit
fruit_Img.frame = CGRect(
    x: 100,
    y: 0,
    width: 100,
    height: 100
```

```
)

// Initialize lost fruit count
fruits_Lost_Num = 0

// Initialize points and update the label
points_Scored = 0

points_Count.text = "Points Scored: 0"

// Add double-tap gesture recognizers to the Restart Button
let restart_button_DoubleTap = UITapGestureRecognizer(
    target: self,
    action: #selector(restart_game_DoubleTap)
)

restart_button_DoubleTap.numberOfTapsRequired = 2

// We added the recognizer to the restart_btn.
restart_btn.addGestureRecognizer(restart_button_DoubleTap)

// Add left and right swipe gesture recognizers to pause_resume_Btn
let pause_button_Swipe = UISwipeGestureRecognizer(
    target: self,
    action: #selector(pause_Game)
)

pause_button_Swipe.direction = .left

pause_resume_Btn.addGestureRecognizer(pause_button_Swipe)

let resume_button_Swipe = UISwipeGestureRecognizer(
    target: self,
    action: #selector(resume_Game)
```

```
    )

    resume_button_Swipe.direction = .right

    pause_resume_Btn.addGestureRecognizer(resume_button_Swipe)
}

@IBAction func game_start(_ sender: UIButton)
{
    // Start game actions

    // Hide start button
    sender.isHidden = true

    // Show container
    container_Img.isHidden = false

    fruit_Img.isHidden = false

    points_Count.isHidden = false

    // Show fruit collected and lost count labels
    fruits_Collected_Count.isHidden = false

    fruits_Lost_Count.isHidden = false

    pause_resume_Btn.isHidden = false

    // Show restart button when the game starts
    restart_btn.isHidden = false

    // Random X position for the Fruit
    fruit_random_X = Int.random(in: 10...300)
```

```
// Set initial position for the Fruit
self.fruit_Img.frame = CGRect(
    x: fruit_random_X,
    y: self.fruit_position_Y,
    width: 40,
    height: 40
)

fruit_Img.image = fruits[Int.random(in: 0...3)]

// Set the initial fruit type
current_fruit_Type = fruits[Int.random(in: 0..<fruits.count)]

fruit_Img.image = current_fruit_Type

// Random X position for the Container
container_position_X = Int.random(in: 10...300)

// Set position and size for the Container
self.container_Img.frame = CGRect(
    x: container_position_X,
    y: 300,
    width: 150,
    height: 80
)

// Calculate timeInterval based on fruitDropSpeed
// Adjust the formula as needed
let fruitDropSpeed_timeInterval = 0.1 / Double(fruit_drop_Speed)

// Start timer to control Fruit dropping
fruit_catching_GameTimer = Timer.scheduledTimer(
    timeInterval: fruitDropSpeed_timeInterval,
    target: self,
```

```
            selector: #selector(fruit_Drop),
            userInfo: nil,
            repeats: true
        )
    }

    func fruit_Hit()
    {
        // Check if the Fruit hits the Container
        if abs(self.fruit_position_Y - 250) < 20 && abs(self.fruit_random_X + 20 - self.container_position_X - Int(self.container_Img.frame.width/2.0)) < 50
        {
            // Reset the game if the Fruit hits the Container
            self.reset_Game()
        }

    }



    @objc func fruit_Drop() {
        // If the game is paused, do nothing
        if is_game_Paused {
            // If isGamePaused is true, the function immediately returns, preventing any further execution and effectively stopping the fruit's movement.
            return
        }

        // Control Fruit dropping and collision detection
        fruit_Hit()

        // Move Fruit downwards
        self.fruit_position_Y = self.fruit_position_Y + 10

        // Update Fruit's X position ONLY based on touch movement if it hasn't been reset yet
```

```
if self.fruit_position_Y > 0 {
    is_fruit_Reset = false
}

// Constrain Fruit's X position within screen bounds
if fruit_random_X > 300
{
    fruit_random_X = 300
}
else if fruit_random_X < 0
{
    fruit_random_X = 0
}

self.fruit_Img.frame = CGRect(
    x: fruit_random_X,
    y: self.fruit_position_Y,
    width: 100,
    height: 100
)

// Reset Fruit's Y position if it goes off-screen
if self.fruit_position_Y >= 500
{
    self.fruit_position_Y = 0

    // Randomize fruit's X position when it's lost
    fruit_random_X = Int.random(in: 10...300)

    self.fruit_Img.frame = CGRect(
        x: fruit_random_X,
        y: self.fruit_position_Y,   // Reset Y position
        width: 100,
        height: 100
```

```
)

// Increment lost fruit count
fruits_Lost_Num = fruits_Lost_Num + 1

// Update fruit lost count label text HERE
fruits_Lost_Count.text = "Fruits Lost: " + String(fruits_Lost_Num)

print("Fruits Lost: ", fruits_Lost_Num)

// Randomize container position when fruit is lost
container_position_X = Int.random(in: 10...300)

self.container_Img.frame = CGRect(
    x: container_position_X,
    y: 300,
    width: 150,
    height: 80
)

// Reset the isFruitReset flag
is_fruit_Reset = false

// Decrement points based on the fruit type
if let pointsForFruit = fruit_Points[current_fruit_Type!]
{
    // Ensure points don't go below 0
    points_Scored = max(0, points_Scored - pointsForFruit)
}
else
{
    // Handle the case where the fruit type is not found in the dictionary (optional)
    print("Error: Fruit type not found in point dictionary")
}
```

```
        points_Count.text = "Points Scored: \(points_Scored)"
    }
}

override func touchesMoved(_ touches: Set<UITouch>, with event: UIEvent?) {
    // Only move the container if the game is not paused
    if !is_game_Paused
    {
        // Handle touch movement to control Container's X position
        let touch1 = touches.first!

        container_position_X = Int(touch1.location(in: self.view).x)

        // Constrain Container's X position within screen bounds (considering its width)
        let maxX = Int(self.view.frame.width - self.container_Img.frame.width)

        container_position_X = max(0, min(maxX, container_position_X))

        // Update Container's position
        self.container_Img.frame.origin.x = CGFloat(container_position_X)
    }
}

func reset_Game()
{
    // Reset game after a Fruit is lost

    // Randomize fruit's X position when it's caught
    fruit_random_X = Int.random(in: 10...300)

    self.fruit_Img.frame = CGRect(
        x: fruit_random_X,
        y: 0,
```

```
    width: 100,
    height: 100
)

// Set the flag after randomizing the X position
is_fruit_Reset = true

// New random X position for the Container
container_position_X = Int.random(in: 10...300)

// Update Container's position
self.container_Img.frame = CGRect(
    x: container_position_X,
    y: 300,
    width: 150,
    height: 80
)

// Increment number of fruits collected
fruits_Collected_Num = fruits_Collected_Num + 1

// Reset Fruit's Y position
self.fruit_position_Y = 0

fruit_Img.image = fruits[Int.random(in: 0...3)]

// Update fruit collected count label text
fruits_Collected_Count.text = "Fruits Collected: " + String(fruits_Collected_Num)

print("Fruits Collected: ", fruits_Collected_Num)

// Increment points based on the fruit type
if let points_ForEachFruit = fruit_Points[current_fruit_Type!]
{
```

```
            points_Scored = points_Scored + points_ForEachFruit
        }
        else
        {
            // Handle the case where the fruit type is not found in the dictionary (optional)
            print("Error: Fruit type not found in points dictionary")
        }

        points_Count.text = "Points Scored: \(points_Scored)"

        /*
         Get a random fruit type
         Update currentFruitType FIRST
        */
        current_fruit_Type = fruits[Int.random(in: 0..<fruits.count)]

        // THEN set the image
        fruit_Img.image = current_fruit_Type
    }

    // We created pauseGame and resumeGame functions to handle the swipe actions.
    @objc func pause_Game(_ sender: UISwipeGestureRecognizer)
    {
        is_game_Paused = true
        fruit_catching_GameTimer?.invalidate()
        fruit_catching_GameTimer = nil
        pause_resume_Btn.setTitle("Resume Game", for: .normal)
    }

    @objc func resume_Game(_ sender: UISwipeGestureRecognizer)
    {
        is_game_Paused = false

        if fruit_catching_GameTimer == nil
```

```
    {
        fruit_catching_GameTimer = Timer.scheduledTimer(
            timeInterval: 0.1,
            target: self,
            selector: #selector(fruit_Drop),
            userInfo: nil,
            repeats: true)
    }

    pause_resume_Btn.setTitle("Pause Game", for: .normal)
}

// We created restart_Game_double_tap function to handle the double-tap actions.
@objc func restart_game_DoubleTap(_ sender: UITapGestureRecognizer)
{
    // Reset counters
    fruits_Collected_Num = 0

    fruits_Lost_Num = 0

    // Reset points and update the label
    points_Scored = 0

    // Update labels
    fruits_Collected_Count.text = "Fruits Collected: 0"

    fruits_Lost_Count.text = "Fruits Lost: 0"

    points_Count.text = "Points Scored: 0"

    // Reset fruit position
    fruit_Img.frame = CGRect(
        x: 100,
        y: 0,
```

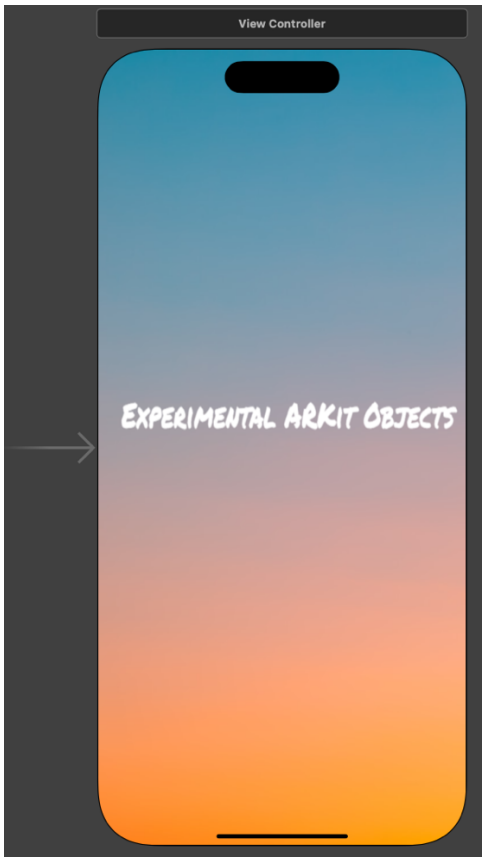
```
    width: 100,
    height: 100
)

// Reset container position
container_position_X = Int.random(in: 10...300)

self.container_Img.frame = CGRect(
    x: container_position_X,
    y: 300,
    width: 150,
    height: 80
)

// If the game was paused, resume it
if is_game_Paused
{
    is_game_Paused = false

    fruit_catching_GameTimer?.invalidate()

    fruit_catching_GameTimer = nil

    fruit_catching_GameTimer = Timer.scheduledTimer(
        timeInterval: 0.1,
        target: self,
        selector: #selector(fruit_Drop),
        userInfo: nil,
        repeats: true
    )

    pause_resume_Btn.setTitle(
        "Pause Game",
        for: .normal
```

```
        )
      }
   }
}
```

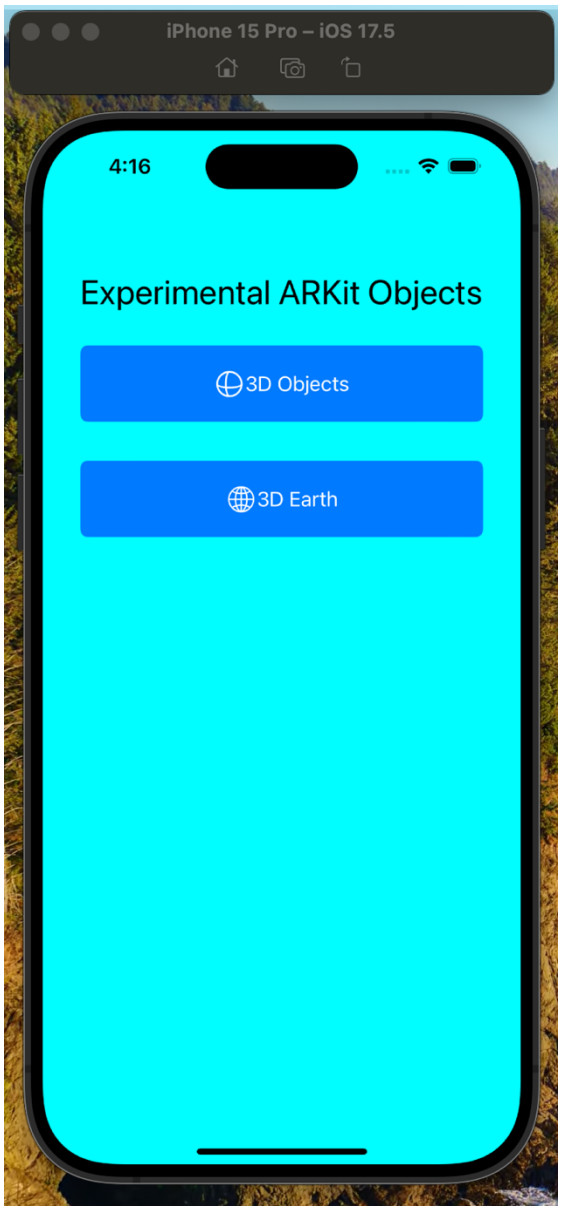
Question 2 (20 marks)

Question 2(a) (10 marks)

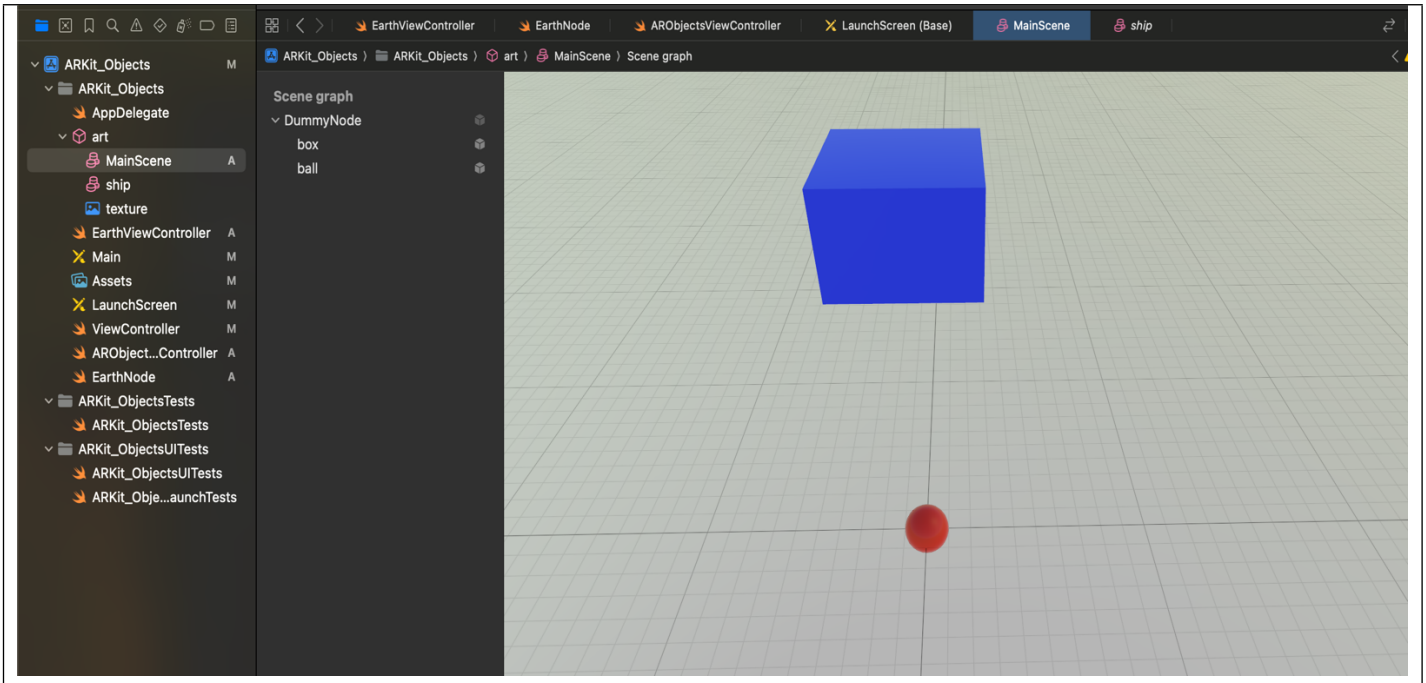
Launch Screen:



Simulator Flow:



Art --> MainScene.scn:



ViewController Source Code:

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var Objects_Button: UIButton!

    @IBOutlet weak var Planes_Button: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

    }

}
```

EarthNode Class Source Code:

```
import SceneKit
```

```
class EarthNode: SCNNode {
    override init() {
        super.init()
        self.geometry = SCNSphere(radius: 0.2)
        self.geometry?.firstMaterial?.diffuse.contents = UIImage(named:"Diffuse")
        self.geometry?.firstMaterial?.specular.contents = UIImage(named:"Specular")
        self.geometry?.firstMaterial?.emission.contents = UIImage(named:"Emission")
        self.geometry?.firstMaterial?.normal.contents = UIImage(named:"Normal")
        self.geometry?.firstMaterial?.isDoubleSided = true

        self.geometry?.firstMaterial?.transparency = 1
        self.geometry?.firstMaterial?.shininess = 50

        let action = SCNAction.rotate(by: 360 * CGFloat((Double.pi)/180), around: SCNVector3(x:0, y:1, z:0), duration: 8)

        let repeatAction = SCNAction.repeatForever(action)

        self.runAction(repeatAction)

    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
    }


}
```

EarthViewController Source Code:

```
import UIKit
import ARKit

class EarthViewController: UIViewController, ARSCNViewDelegate {
```

```
@IBOutlet var sceneView: ARSCNView!

override func viewDidLoad() {
    super.viewDidLoad()

    // Set the view's delegate
    sceneView.delegate = self

    // Show statistics such as fps and timing information
    sceneView.showsStatistics = true

    let scene = SCNScene()

    // Set the scene to the view
    sceneView.scene = scene
}



override func viewWillAppear(_ animated: Bool) {
    super.viewWillAppear(animated)

    // Create a session configuration
    let configuration = ARWorldTrackingConfiguration()

    // Run the view's session
    sceneView.session.run(configuration)
}

override func viewWillDisappear(_ animated: Bool) {
    super.viewWillDisappear(animated)

    // Pause the view's session
    sceneView.session.pause()
```

```
    }

    override func touchesBegan(_ touches: Set<UITouch>, with event: UIEvent?)
    {
        let touch = touches.first
        let location = touch?.location(in: sceneView)

        let hitResults = sceneView.hitTest(location!, types: .featurePoint)

        if let hitTestResult = hitResults.first
        {
            let transform = hitTestResult.worldTransform
            let position = SCNVector3(x: transform.columns.3.x, y: transform.columns.3.y, z:transform.columns.3.z)

            let newEarth = EarthNode()
            newEarth.position = position

            sceneView.scene.rootNode.addChildNode(newEarth)
        }
    }

    // MARK: - ARSCNViewDelegate

/*
    // Override to create and configure nodes for anchors added to the view's session.
    func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
        let node = SCNNode()

        return node
    }
*/

    func session(_ session: ARSession, didFailWithError error: Error) {
        // Present an error message to the user
```

```
    }

    func sessionWasInterrupted(_ session: ARSession) {
        // Inform the user that the session has been interrupted, for example, by presenting an overlay

    }

    func sessionInterruptionEnded(_ session: ARSession) {
        // Reset tracking and/or remove existing anchors if consistent tracking is required

    }

}
```

ARObjectsViewController Source Code:

```
import UIKit
import SceneKit
import ARKit

class ARObjectsViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    var ball = SCNNode()

    var box = SCNNode()

    override func viewDidLoad() {
        super.viewDidLoad()

        // Set the view's delegate
        sceneView.delegate = self
```

```
        // Show statistics such as fps and timing information
        sceneView.showsStatistics = true

        sceneView.allowsCameraControl = true

        // Create a new scene
        let scene = SCNScene(named: "art.scnassets/MainScene.scn")!

        // Set the scene to the view
        sceneView.scene = scene

        let wait:SCNAction = SCNAction.wait(duration: 3)

        let runAfter:SCNAction = SCNAction.run { _ in self.addSceneContent()
        }

        let seq:SCNAction = SCNAction.sequence( [wait, runAfter] )

        sceneView.scene.rootNode.runAction(seq)

        let tapGestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(handleTap(sender:)))

        self.sceneView.addGestureRecognizer(tapGestureRecognizer)
    }


    @objc func handleTap(sender: UITapGestureRecognizer)
    {
//        guard let sceneView = sender.view as? ARSCNView else {
//            return
//        }

        let touchLocation = sender.location(in: sceneView)
```

```
        let hitTestResult = sceneView.hitTest(touchLocation, options: [:])

        if !hitTestResult.isEmpty {

            for hitResult in hitTestResult
            {
//                print(hitTestResult.node.name)

                if (hitResult.node == box) {
                    box.physicsBody?.applyForce(SCNVector3(0, 10, 3), asImpulse: true)
                }
            }
        }
    }


    func addSceneContent()
    {
        let dummyNode = self.sceneView.scene.rootNode.childNode(withName: "DummyNode", recursively: false)

        dummyNode?.position = SCNVector3(0, -5, -5)

        self.sceneView.scene.rootNode.enumerateChildNodes { (node, _) in

            if (node.name == "box")
            {
                print("found box")

                box = node

                box.physicsBody = SCNPhysicsBody(type: .dynamic, shape: SCNPhysicsShape(node: ball, options: nil))

                box.physicsBody?.isAffectedByGravity = true
```

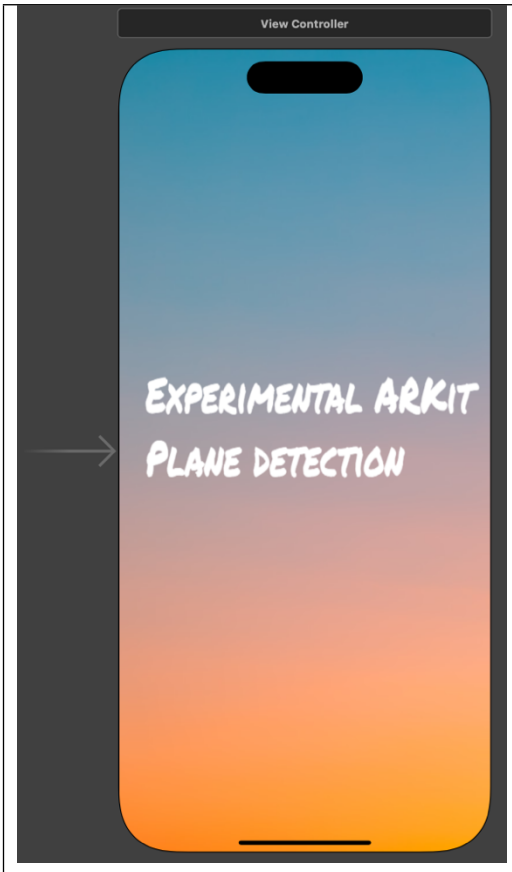
```
            box.physicsBody?.restitution = 1
        }
        else if (node.name == "ball")
        {
            print("found box")

            ball = node

            let ballGeometry = ball.geometry

            let ballShape: SCNPhysicsShape = SCNPhysicsShape(geometry: ballGeometry!, options: nil)

            ball.physicsBody = SCNPhysicsBody(type: .static, shape: ballShape)

            ball.physicsBody?.restitution = 1

        }

    }


    let light = SCNLight()

    light.type = SCNLight.LightType.omni

    let lightNode = SCNNode()

    lightNode.light = light

    lightNode.position = SCNVector3(x: 1.6, y: 1.5, z: 1.5)

    self.sceneView.scene.rootNode.addChildNode(lightNode)
}
```

```
    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()

        self.sceneView.debugOptions = [ARSCNDebugOptions.showWorldOrigin, ARSCNDebugOptions.showFeaturePoints, SCNDebugOptions.showPhysicsShapes]



        // Run the view's session
        sceneView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // Pause the view's session
        sceneView.session.pause()
    }

    // MARK: - ARSCNViewDelegate

/*
    // Override to create and configure nodes for anchors added to the view's session.
    func renderer(_ renderer: SCNSceneRenderer, nodeFor anchor: ARAnchor) -> SCNNode? {
        let node = SCNNode()

        return node
    }
*/
```

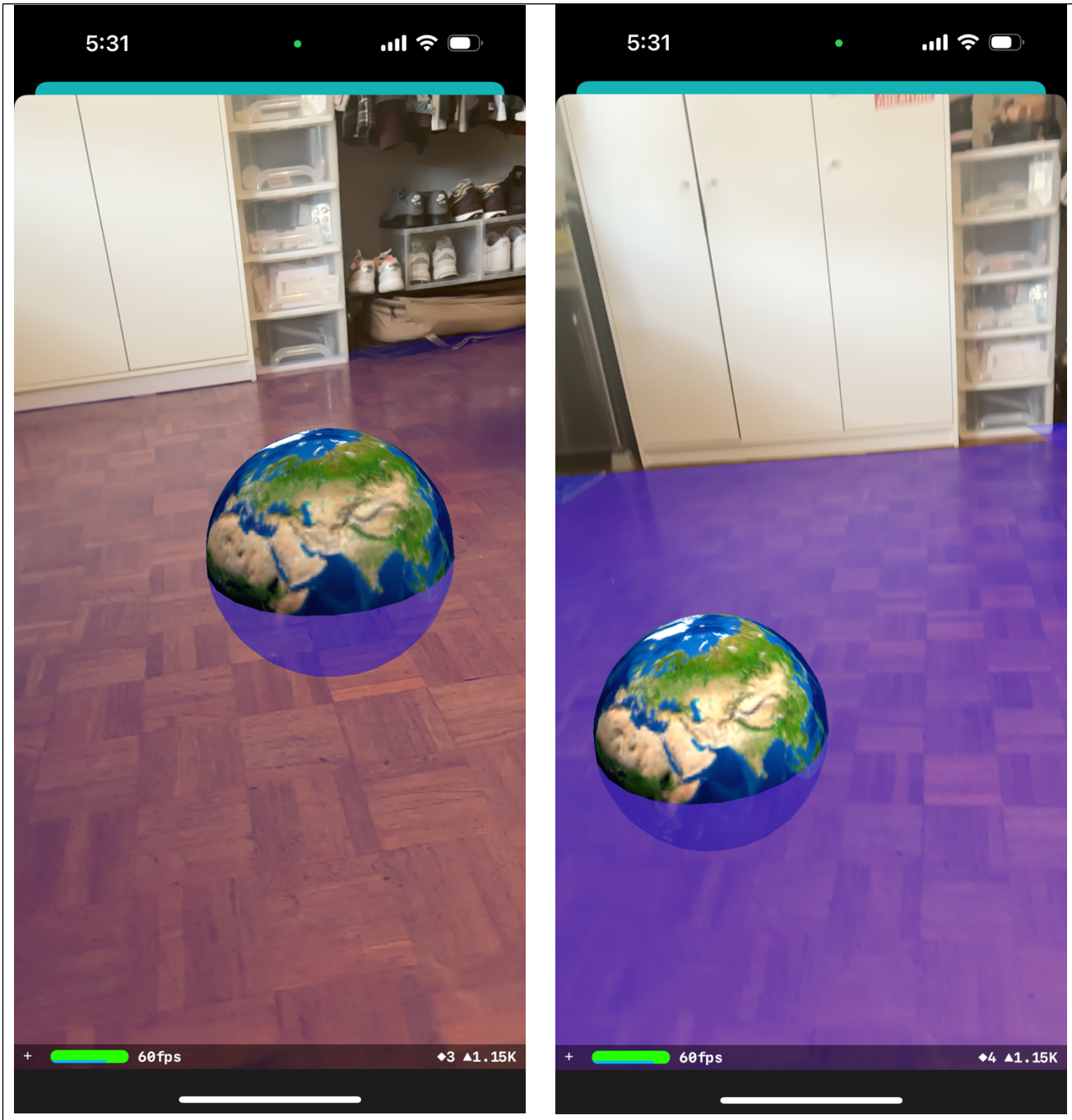
```
    func session(_ session: ARSession, didFailWithError error: Error) {
        // Present an error message to the user

    }

    func sessionWasInterrupted(_ session: ARSession) {
        // Inform the user that the session has been interrupted, for example, by presenting an overlay

    }

    func sessionInterruptionEnded(_ session: ARSession) {
        // Reset tracking and/or remove existing anchors if consistent tracking is required

    }
}
```

Question 2(b) (10 marks)

Launch Screen:



iPhone Flow:



View Controller Source Code:

```
import UIKit

class ViewController: UIViewController {

    @IBOutlet weak var Home_Label: UILabel!
```

```
    @IBOutlet weak var HorizontalPD_Btn: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }

}
```

Earth Node Class Code:

```
import SceneKit

class EarthNode: SCNNode
{
    override init()
    {
        super.init()

        self.geometry = SCNSphere(radius: 0.2)

        self.geometry?.firstMaterial?.diffuse.contents = UIImage(named:"Diffuse")

        self.geometry?.firstMaterial?.specular.contents = UIImage(named:"Specular")

        self.geometry?.firstMaterial?.emission.contents = UIImage(named:"Emission")

        self.geometry?.firstMaterial?.normal.contents = UIImage(named:"Normal")

        self.geometry?.firstMaterial?.isDoubleSided = true

        self.geometry?.firstMaterial?.transparency = 1
```

```
        self.geometry?.firstMaterial?.shininess = 50

        let action = SCNAction.rotate(by: 360 * CGFloat((Double.pi)/180), around: SCNVector3(x:0, y:1, z:0), duration: 8)

        let repeatAction = SCNAction.repeatForever(action)

        self.runAction(repeatAction)

    }

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
    }


}
```

HorizontalPDViewController Source Code:

```
import UIKit
import SceneKit
import ARKit

class HorizontalPDViewController: UIViewController, ARSCNViewDelegate {

    @IBOutlet var sceneView: ARSCNView!

    override func viewDidLoad() {
        super.viewDidLoad()

        // Set the view's delegate
        sceneView.delegate = self

        // Show statistics such as fps and timing information
```

```
        sceneView.showsStatistics = true

        sceneView.allowsCameraControl = true

        let scene = SCNScene()

        // Set the scene to the view
        sceneView.scene = scene


    }

    override func viewWillAppear(_ animated: Bool) {
        super.viewWillAppear(animated)

        // Create a session configuration
        let configuration = ARWorldTrackingConfiguration()

        configuration.planeDetection = .horizontal


        // Run the view's session
        sceneView.session.run(configuration)
    }

    override func viewWillDisappear(_ animated: Bool) {
        super.viewWillDisappear(animated)

        // Pause the view's session
        sceneView.session.pause()
    }

    func session(_ session: ARSession, didFailWithError error: Error) {
        // Present an error message to the user
```

```
}

func sessionWasInterrupted(_ session: ARSession) {
    // Inform the user that the session has been interrupted, for example, by presenting an overlay

}

func sessionInterruptionEnded(_ session: ARSession) {
    // Reset tracking and/or remove existing anchors if consistent tracking is required

}

func renderer(_ renderer: any SCNSceneRenderer, didAdd node: SCNNode, for anchor: ARAnchor)
{
    guard let planeAnchor = anchor as? ARPlaneAnchor else {
        return
    }

    let width = CGFloat(planeAnchor.planeExtent.width)

    let height = CGFloat(planeAnchor.planeExtent.height)

    let plane = SCNPlane(width: width, height: height)

    plane.materials.first?.diffuse.contents = UIColor.blue.withAlphaComponent(0.5)

    let planeNode = SCNNode(geometry: plane)

    let x = CGFloat(planeAnchor.center.x)

    let y = CGFloat(planeAnchor.center.y)

    let z = CGFloat(planeAnchor.center.z)
```

```
    planeNode.position = SCNVector3(x, y, z)

    planeNode.eulerAngles.x = -.pi / 2

    node.addChildNode(planeNode)

    print("Plane Detected")

    let newEarth = EarthNode()

    // Place the newEarth at the center of the detected plane
    newEarth.position = SCNVector3(planeAnchor.center.x, planeAnchor.center.y, planeAnchor.center.z)

    // Add the newEarth to the scene
    node.addChildNode(newEarth)

    // Add a light source to illuminate the newEarth
    let light = SCNLight()

    light.type = SCNLight.LightType.omni

    let lightNode = SCNNode()

    lightNode.light = light

    // Position above the newEarth
    lightNode.position = SCNVector3(x: 0, y: 1, z: 0)

    node.addChildNode(lightNode)

    print("Plane Detected and EarthNode added")

}
```

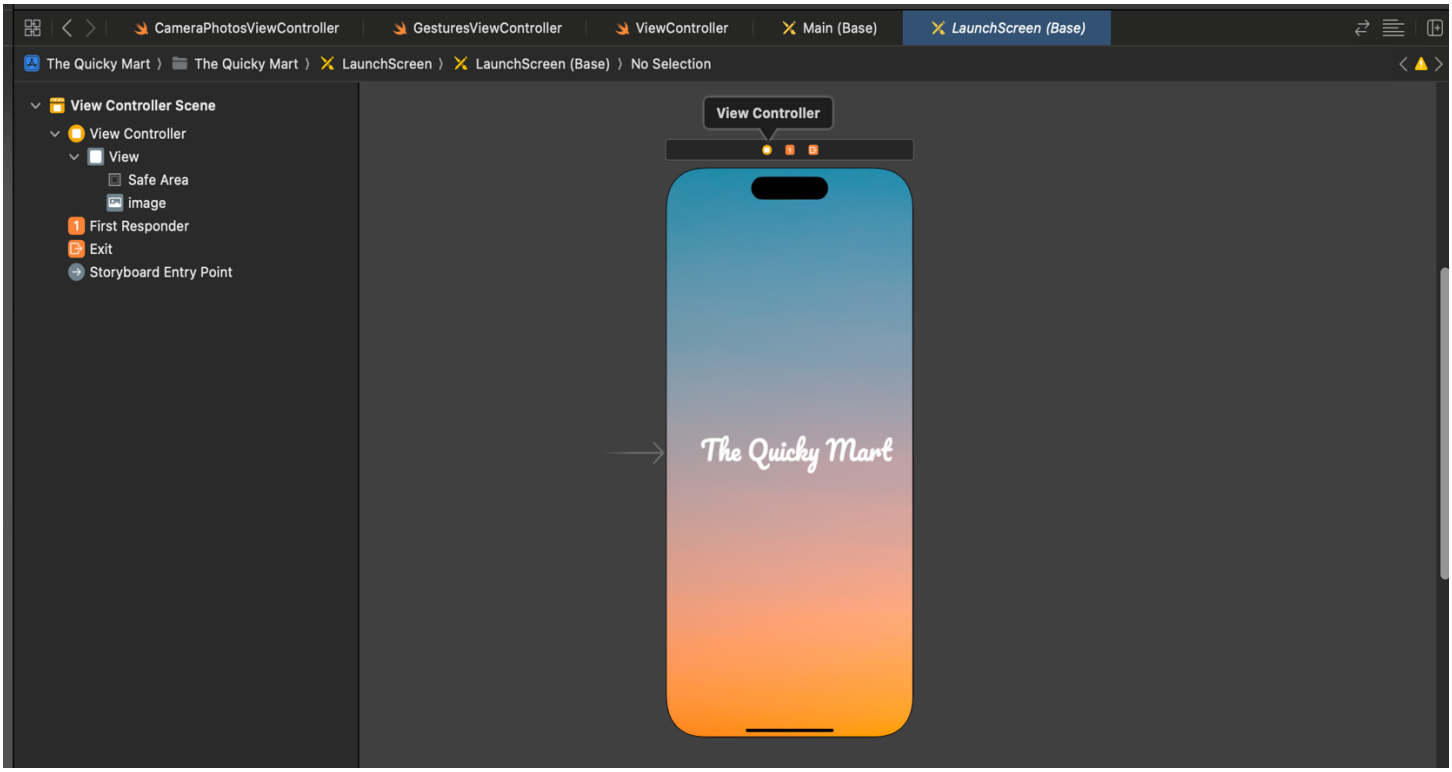
```
func renderer(_ renderer: any SCNSceneRenderer, didUpdate node: SCNNode, for anchor: ARAnchor)
{
    guard let planeAnchor = anchor as? ARPlaneAnchor else {
        return
    }

    print("Plane Updated")

    let planeNode = node.childNodes.first

    guard let plane = planeNode?.geometry as? SCNPlane else {return}

    let width = CGFloat(planeAnchor.planeExtent.width)

    let height = CGFloat(planeAnchor.planeExtent.height)

    plane.width = width

    plane.height = height

    let x = CGFloat(planeAnchor.center.x)

    let y = CGFloat(planeAnchor.center.y)

    let z = CGFloat(planeAnchor.center.z)

    planeNode!.position = SCNVector3(x, y, z)
}


@objc func handlePlusButtonTapped() {
    print("Tapped on plus Button")
```

```
        addSceneContent()
    }

    func addSceneContent() {
        let boxNode = SCNNode()

        boxNode.geometry = SCNBox(width: 0.05, height: 0.05, length: 0.05, chamferRadius: 0.0)

        boxNode.position = SCNVector3(x: 0, y: 0, z: 0)

        self.sceneView.scene.rootNode.addChildNode(boxNode)
    }
}
```

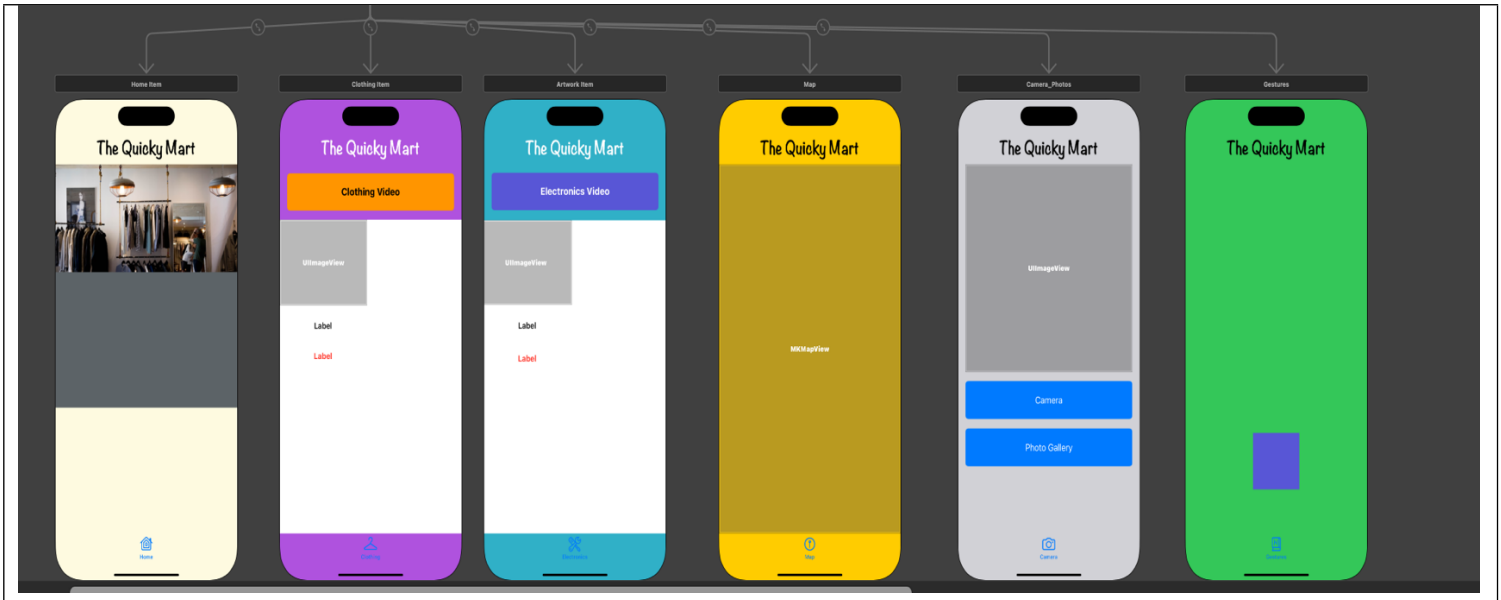
Question 3

Question 3(a) (10 marks)

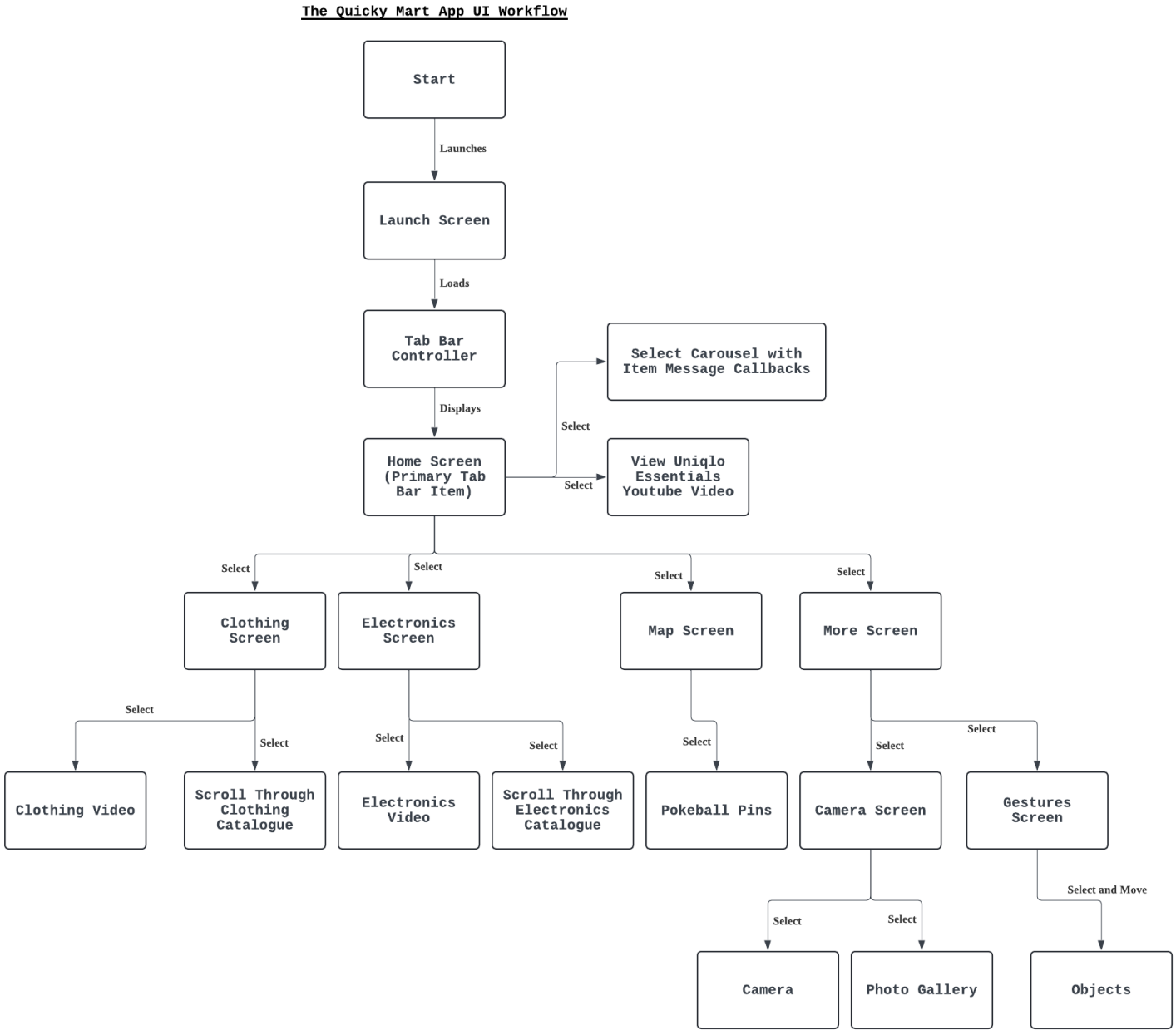
LaunchScreen:



The Quicky Mart App Workflow in Xcode:



The Quicky Mart App Workflow Flowchart



Explanation of the Concepts and Functions of The Quicky Mart App Design

I developed an called The Quicky Mart App, it has a Launch Screen image that will be the first object that people will see in the app. After seeing the Launch Screen, users will be redirected to the Home Screen, which is 1 of several Tab Bar Items that is controlled by a Tab Bar View Controller. Each of the Tab Bar Items are View Controllers that are connected to the Tab Bar View Controller using a Relationship segue.

In the Home Screen, users can watch a Uniqlo clothing YouTube video and interact with a carousel that has 3 squares, when a square is selected, the users will be displayed with a callback message which can be dismissed.

When the user selects on the Clothing Tab Bar Item, they will be directed to a Clothing screen that has a Clothing video and a clothing catalogue that display the image, name and price of each item. The

Clothing catalogue was created using a UICollectionView that has several UICollectionViewCell's. The data structure struct is used to store each clothing title, price and image and the Clothing struct takes its data from an array called clothings.

When the user selects on the Electronics Tab Bar Item, they will be directed to a Electronics screen that has a Electronics video and a electronic catalogue that display the image, name and price of each item. The Electronics catalogue was created using a UICollectionView that has several UICollectionViewCell's. The data structure struct is used to store each electronic title, price and image and the Electronic struct takes its data from an array called electronics.

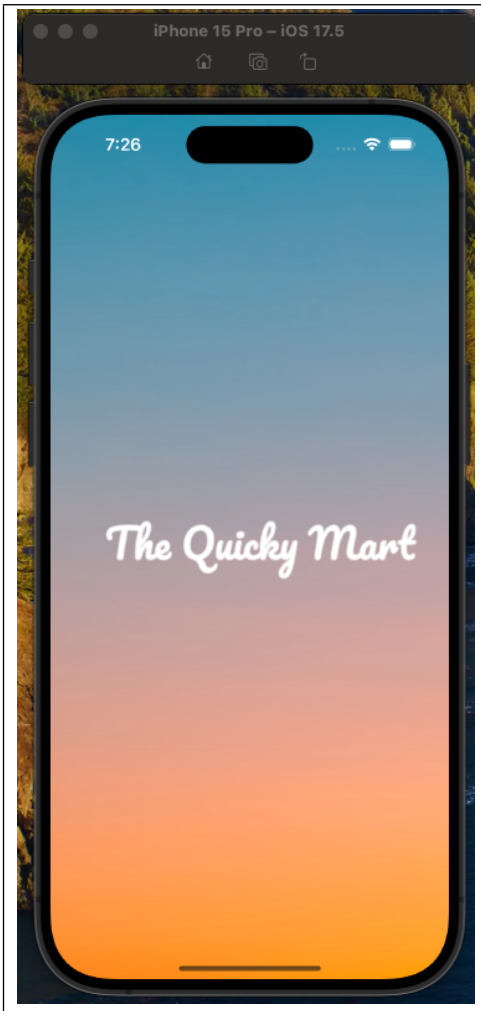
When the user selects on the Map Tab Bar Item, they will be directed to a Map screen where they can select on several custom pins that are represented by Pokeballs and when users select on each of these Pokeballs they will be displayed with a meesage that has a title and subtitle.

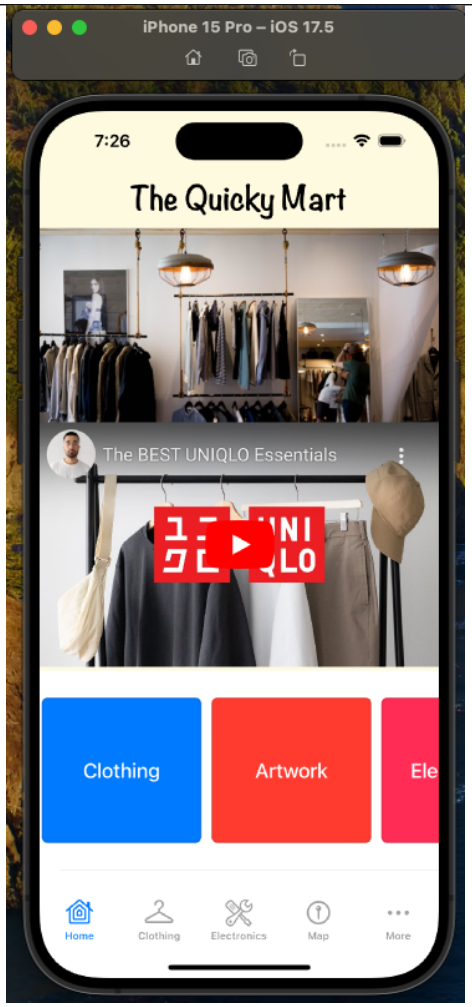
When the user selects on the More Tab Bar Item, they will be directed to a More screen where they will view a list of Tab Bar Items that contains the Camera and Gestures. When users select on the Tab Bar Item Camera, they will be directed to a Camera screen where there is a blank Image View and a camera and photo gallery button. The camera button allows users to use their Phone camera to take photos and upload photos to the app. The photo gallery button allows to upload photos from their Phone photo gallery to the app.

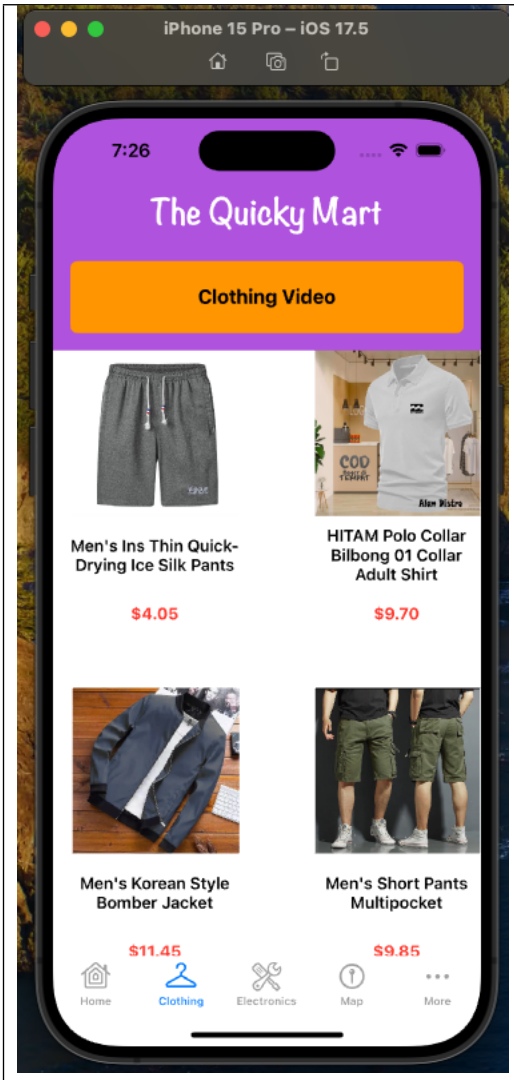
When users select on the Tab Bar Item Gestures. they will be directed to a Gestures screen where there a large square and a small square. When a user selects on the large square, its color automatically changes. Meanwhile for the smaller square, users are able to move the square around the screen. When users long presses on the small square, notice how the square would become larger and revert to its original size once the users stops long pressing on the square. These features demonstrates the usage of several types of gesture recognizers in The Quicky Mart App.

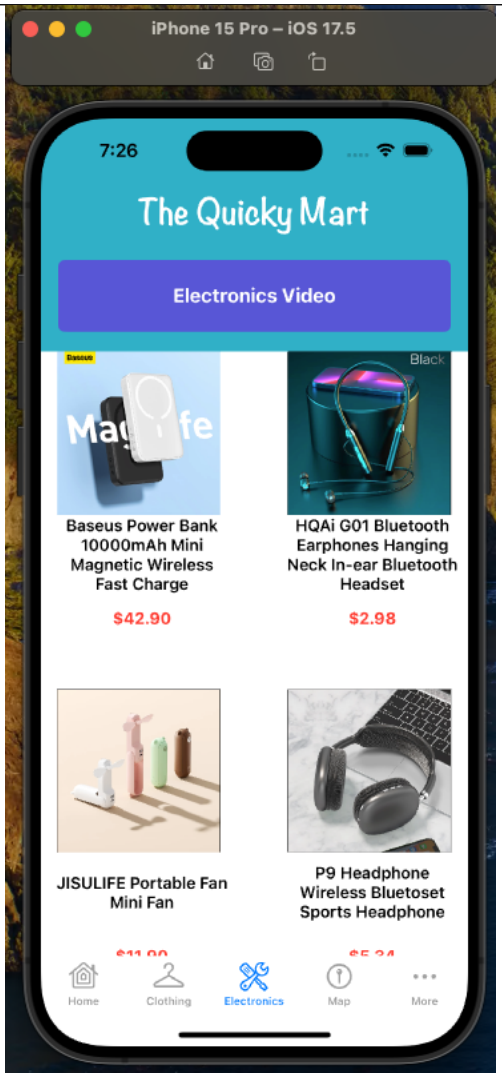
Question 3(b) (30 marks)

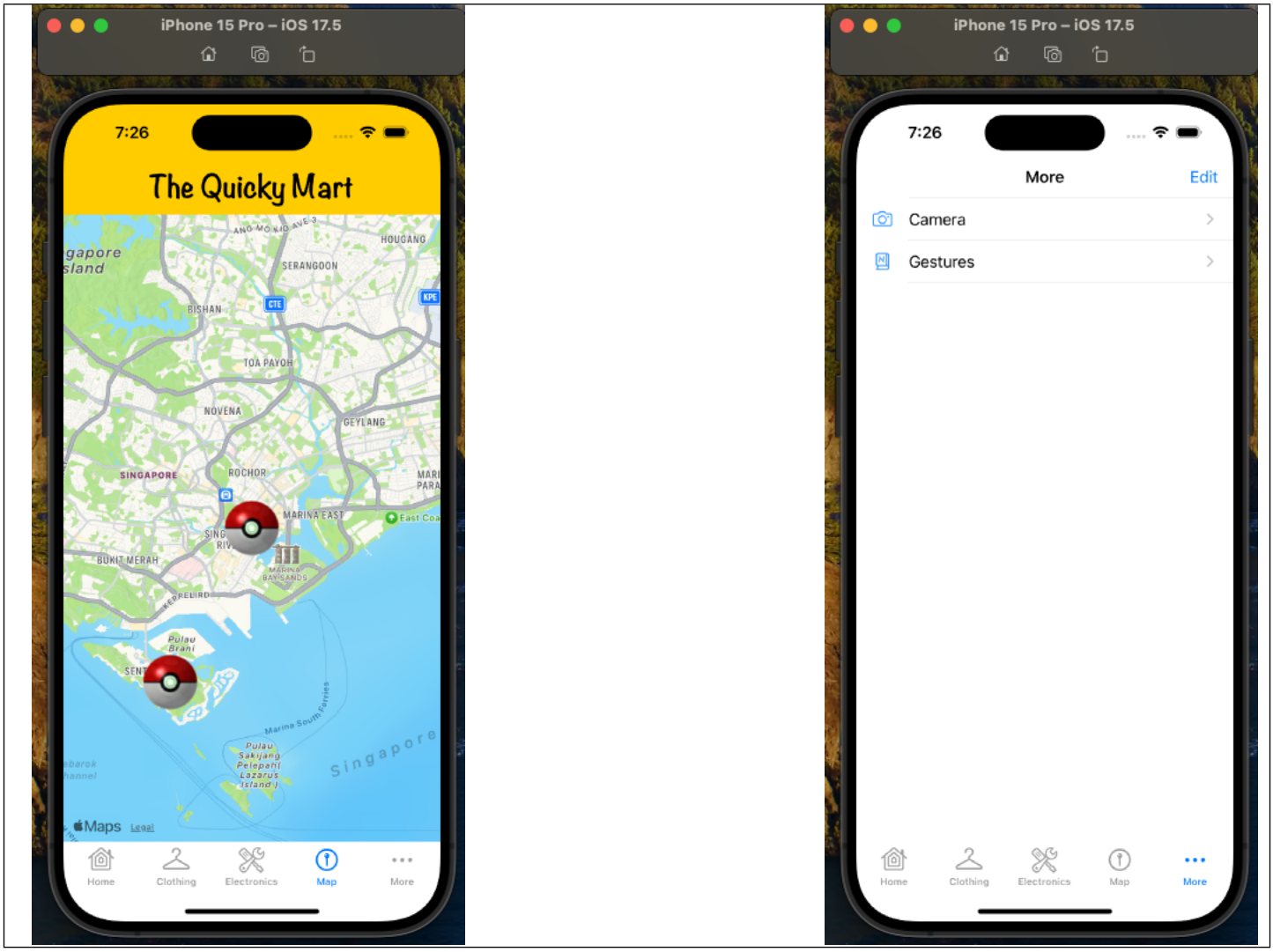
Simulator Flow:

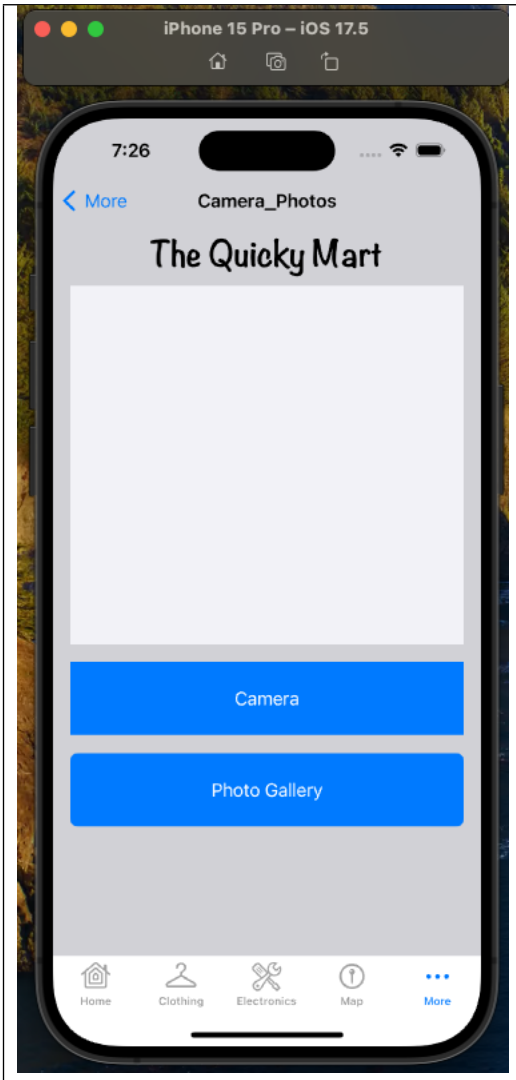

iPhone 15 Pro – iOS 17.5
7:26
The Quicky Mart

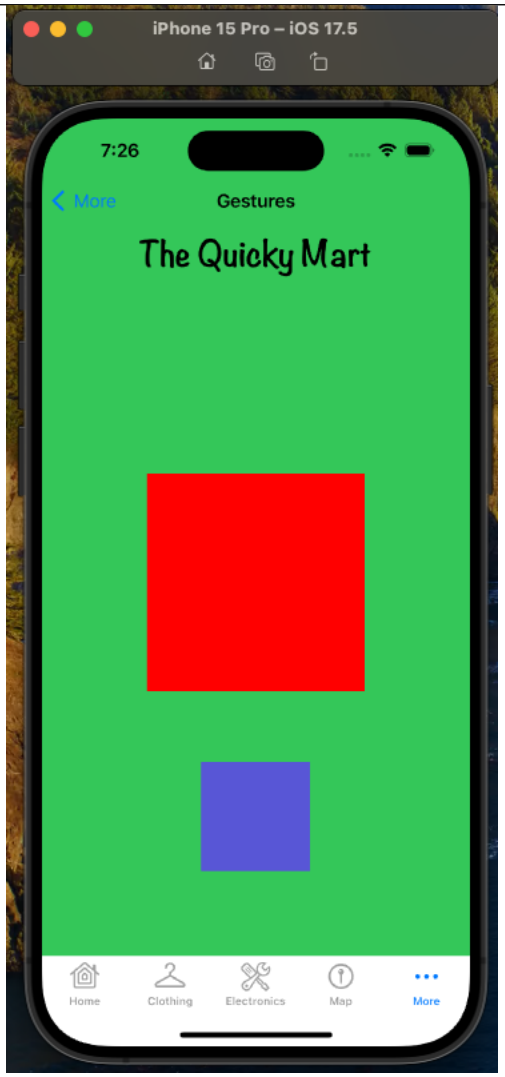

iPhone 15 Pro – iOS 17.5
7:26
The Quicky Mart
The BEST UNIQLO Essentials
Clothing
Artwork
Home
Clothing
Electronics
Map
More


iPhone 15 Pro – iOS 17.5
7:26
The Quicky Mart
Clothing Video
Men's Ins Thin Quick-Drying Ice Silk Pants
$4.05
HITAM Polo Collar Bilbong 01 Collar Adult Shirt
$9.70
Men's Korean Style Bomber Jacket
$11.45
Men's Short Pants Multipocket
$9.85
Home
Clothing
Electronics
Map
More


iPhone 15 Pro – iOS 17.5
7:26
The Quicky Mart
Electronics Video
Baseus Power Bank 10000mAh Mini Magnetic Wireless Fast Charge
$42.90
HQAi G01 Bluetooth Earphones Hanging Neck In-ear Bluetooth Headset
$2.98
JISULIFE Portable Fan Mini Fan
P9 Headphone Wireless Bluetoset Sports Headphone
Home
Clothing
Electronics
Map
More


iPhone 15 Pro – iOS 17.5
7:26
The Quicky Mart
SERANGOON
HOUGANG
BISHAN
TOA PAYOH
NOVENA
GEYLANG
SINGAPORE
ROCHOR
MARINA EAST
BUKIT MERAH
Maps
Legal
Home
Clothing
Electronics
Map
More
iPhone 15 Pro – iOS 17.5
7:26
More
Edit
Camera
Gestures
Home
Clothing
Electronics
Map
More





ViewController Source Code:

```
//
//  ViewController.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 11/9/24.
//

import UIKit

// used for create AV Player View -- acceleration audio, forward...
import AVKit
```

```
// Used for displaying web content
import WebKit

class ViewController: UIViewController, UITableViewDelegate, UITableViewDataSource
{

    private let tableView: UITableView = {
        let table = UITableView()

        table.register(CollectionTableViewCell.self, forCellReuseIdentifier: CollectionTableViewCell.identifier)

        return table
    }()


    private let viewModels: [CollectionTableViewCellViewModel] = [

        CollectionTableViewCellViewModel(

            viewModels: [

                TileCollectionViewCellViewModel(name: "Clothing", backgroundColor: .systemBlue),
                TileCollectionViewCellViewModel(name: "Artwork", backgroundColor: .systemRed),
                TileCollectionViewCellViewModel(name: "Electronics", backgroundColor: .systemPink)

            ]
        )

    ]

    @IBOutlet weak var homeVideo: WKWebView!

    override func viewDidLoad() {
        super.viewDidLoad()
```

```
        // Do any additional setup after loading the view.


        getVideo(videoCode: "K2HS3_8cTx8")

        view.addSubview(tableView)

        tableView.dataSource = self

        tableView.delegate = self

    }



    func getVideo(videoCode: String)
    {
        let url = URL(string: "https://www.youtube.com/embed/\(videoCode)")

        homeVideo.load(URLRequest(url: url!))
    }



    override func viewDidLayoutSubviews()
    {
        super.viewDidLayoutSubviews()

//        tableView.frame = view.bounds

        // Calculate the frame for the tableView at the bottom
        let tableViewHeight = view.frame.size.width / 1.3 // Assuming height is half the screen width
        let tableViewY = view.bounds.height - tableViewHeight
        tableView.frame = CGRect(x: 0, y: tableViewY, width: view.bounds.width, height: tableViewHeight)
    }
```

```
    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int) -> Int
    {
        return viewModels.count
    }


    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell
    {
        let viewModel = viewModels[indexPath.row]

        guard let cell = tableView.dequeueReusableCell(withIdentifier: CollectionTableViewCell.identifier, for: indexPath) as? CollectionTableViewCell else {
            fatalError()
        }

        cell.delegate = self

        cell.configure(with: viewModel)

//        cell.textLabel?.text = "Hello World"

        return cell
    }


    func tableView(_ tableView: UITableView, heightForRowAt indexPath: IndexPath) -> CGFloat
    {
        return view.frame.size.width/2
    }
```

```
}



extension ViewController: CollectionTableViewCellDelegate {
    func collectionTableViewCellDidTapItem(with viewModel: TileCollectionViewCellViewModel) {
        let alert = UIAlertController(title: viewModel.name, message: "You successfully got the selected item!", preferredStyle: .alert)


        alert.addAction(UIAlertAction(title: "Dismiss", style: .cancel, handler: nil))


        present(alert, animated: true)
    }
}
```

CollectionTableViewCell Class Source Code:

```
//
//  CollectionTableViewCell.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 11/9/24.
//


import UIKit


struct CollectionTableViewCellViewModel {
    let viewModels: [TileCollectionViewCellViewModel]
}


protocol CollectionTableViewCellDelegate: AnyObject {
    func collectionTableViewCellDidTapItem(with viewModel: TileCollectionViewCellViewModel)
}


class CollectionTableViewCell: UITableViewCell, UICollectionViewDelegate, UICollectionViewDataSource, UICollectionViewDelegateFlowLayout
```

```
{

//
//    override func awakeFromNib() {
//        super.awakeFromNib()
//        // Initialization code
//    }
//
//    override func setSelected(_ selected: Bool, animated: Bool) {
//        super.setSelected(selected, animated: animated)
//
//        // Configure the view for the selected state
//    }

    static let identifier = "CollectionTableViewCell"

    weak var delegate: CollectionTableViewCellDelegate?

    private var viewModels: [TileCollectionViewCellViewModel] = []



    private let collectionView: UICollectionView = {
        let layout = UICollectionViewFlowLayout()

        layout.scrollDirection = .horizontal

        layout.sectionInset = UIEdgeInsets(top: 2, left: 2, bottom: 2, right: 2)

        let collectionView = UICollectionView(
            frame: .zero,
            collectionViewLayout: layout
        )
```

```
        collectionView.register(
            TileCollectionViewCell.self,
            forCellWithReuseIdentifier: TileCollectionViewCell.identifier
        )

        collectionView.backgroundColor = .systemBackground

        return collectionView
    }()

    // MARK - Init

    override init(style: UITableViewCell.CellStyle, reuseIdentifier: String?)
    {
        super.init(style: style, reuseIdentifier: reuseIdentifier)

        contentView.backgroundColor = .systemBackground

        contentView.addSubview(collectionView)

        collectionView.delegate = self

        collectionView.dataSource = self
    }

    required init?(coder: NSCoder) {
        fatalError()
    }


    // MARK: - Layout

    override func layoutSubviews()
    {
```

```
        super.layoutSubviews()

        collectionView.frame = contentView.bounds
    }

    // MARK - CollectionView

    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int
    {
        return viewModels.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell
    {
        guard let cell = collectionView.dequeueReusableCell(
            withReuseIdentifier: TileCollectionViewCell.identifier,
            for: indexPath
        ) as? TileCollectionViewCell else {
            fatalError()
        }
        cell.configure(with: viewModels[indexPath.row])
        return cell
    }

    func configure(with viewModel: CollectionTableViewCellViewModel) {
        self.viewModels = viewModel.viewModels
        collectionView.reloadData()
    }

    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
sizeForItemAt indexPath: IndexPath) -> CGSize
    {
        let width: CGFloat = contentView.frame.size.width/2.5
```

```
        return CGSize(width: width, height: width/1.1)
    }

    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath)
    {
        collectionView.deselectItem(at: indexPath, animated: true)
        let viewModel = viewModels[indexPath.row]
        delegate?.collectionTableViewCellDidTapItem(with: viewModel)
    }
}
```

TileCollectionViewCell Class Source Code:

```
//
//  TileCollectionViewCell.swift
//  Carousel
//
//  Created by Shawn Yang on 11/9/24.
//
import UIKit
struct TileCollectionViewCellViewModel {
    let name: String
    let backgroundColor: UIColor
}

class TileCollectionViewCell: UICollectionViewCell
{
    static let identifier =  "TileCollectionViewCell"
    private let label: UILabel = {
        let label = UILabel()
        label.textColor = .white
        label.textAlignment = .center
        label.font = .systemFont(ofSize: 20, weight: .medium)
        return label
```

```
    }()

    override init(frame: CGRect) {
        super.init(frame: frame)
        contentView.addSubview(label)
        contentView.layer.cornerRadius = 6
        contentView.layer.borderWidth = 1.5
        contentView.layer.borderColor = UIColor.quaternaryLabel.cgColor
    }

    required init?(coder: NSCoder) {
        fatalError()
    }

    override func layoutSubviews()
    {
        super.layoutSubviews()
        label.frame = contentView.bounds
    }

    func configure(with viewModel: TileCollectionViewCellViewModel)
    {
        contentView.backgroundColor = viewModel.backgroundColor
        label.text = viewModel.name
    }
}
```

ClothingViewController Source Code:

```
//
//  ClothingViewController.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 12/9/24.
```

```
//
import UIKit
// used to create audio video players and play
import AVFoundation
// used for create AV Player View -- acceleration audio, forward...
import AVKit
class ClothingViewController: UIViewController {
    @IBOutlet weak var collectionView: UICollectionView!
    let file_vid_url = Bundle.main.url(forResource: "FallOutfitsForMen", withExtension: "mov")
    // Video player object
    var Vidplayer: AVPlayer!
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        // Initialize the video player
        if let path = file_vid_url
        {
            // Try to create a video player with the provided URL
            Vidplayer = try! AVPlayer(url: path)
        }
        else
        {
            // Print an error message if the video file is not found
            print("Video file not found")
        }
        collectionView.dataSource = self
        collectionView.delegate = self
        collectionView.collectionViewLayout = UICollectionViewFlowLayout()
    }


    @IBAction func playClothingVideo(_ sender: UIButton) {
        // Create an AVPlayerViewController to display the video
        let AVcontroller = AVPlayerViewController()
```

```
        // Set the video player for the AVPlayerViewController
        AVcontroller.player = Vidplayer
        // Present the AVPlayerViewController and start playing the video
        present(AVcontroller, animated: true)
        {
            AVcontroller.player!.play()
        }
    }

    /*
    // MARK: - Navigation
    // In a storyboard-based application, you will often want to do a little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?) {
        // Get the new view controller using segue.destination.
        // Pass the selected object to the new view controller.
    }
    */
}

extension ClothingViewController: UICollectionViewDataSource {
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return clothings.count
    }

    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell
{
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "ClothingCollectionViewCell", for: indexPath)
as! ClothingCollectionViewCell
        cell.setup(with: clothings[indexPath.row])
        return cell
    }
}

extension ClothingViewController: UICollectionViewDelegateFlowLayout {
```

```
    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout, sizeForItemAt indexPath: IndexPath) -> CGSize {
        return CGSize(width: 180, height: 300)
    }
}

extension ClothingViewController: UICollectionViewDelegate {
    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        print(clothings[indexPath.row].title)
    }
}
```

Clothing Source Code:

```
import UIKit
struct Clothing {
    let title: String
    let price: String
    let image: UIImage
}
let clothings: [Clothing] = [
    Clothing(title: "Men's Ins Thin Quick-Drying Ice Silk Pants", price: "$4.05" , image:  imageLiteral(resourceName: " Men's Ins Thin Quick-Drying Ice Silk Pants  ($4.05)")),
    Clothing(title: "HITAM Polo Collar Bilbong 01 Collar Adult Shirt", price: "$9.70" , image:  imageLiteral(resourceName: "HITAM Polo Collar Bilbong 01 Collar Adult Shirt ($9.70)")),
    Clothing(title: "Men's Korean Style Bomber Jacket", price: "$11.45" , image:  imageLiteral(resourceName: "Men's Korean Style Bomber Jacket ($11.45)")),
    Clothing(title: "Men's Short Pants Multipocket", price: "$9.85" , image:  imageLiteral(resourceName: "Men's Short Pants Multipocket ($9.85)")),
    Clothing(title: "Men's Shorts Summer Stretch Ice Silk Quick-Drying Beach Pants", price: "$5.16" , image: imageLiteral(resourceName: "Men's Shorts Summer Stretch Ice Silk Quick-Drying Beach Pants ($5.16)")),
    Clothing(title: "Men's Trousers", price: "$6.43" , image:  imageLiteral(resourceName: "Men's Trousers ($6.43)")),
    Clothing(title: "Polo T Shir Men Shirt Zipper 100% Cotton", price: "$11.77" , image:  imageLiteral(resourceName: "Polo T Shir Men Shirt Zipper 100% Cotton ($11.77)")),
    Clothing(title: "Unisex loose-fitting short-sleeved T-shirt with WHEN EVER lettering", price: "$4.25" , image: imageLiteral(resourceName: "Unisex loose-fitting short-sleeved T-shirt with WHEN EVER lettering ($4.25)"))
]
```

ClothingCollectionViewCell Class Source Code:

```
//
//  MovieCollectionViewCell.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 12/9/24.
//
import UIKit
class ClothingCollectionViewCell: UICollectionViewCell {
    @IBOutlet weak var clothingImageView: UIImageView!
    @IBOutlet weak var clothingTitleLabel: UILabel!
    @IBOutlet weak var clothingPriceLabel: UILabel!

    func setup(with clothing: Clothing) {
        clothingImageView.image = clothing.image
        clothingTitleLabel.text = clothing.title
        clothingPriceLabel.text = clothing.price
    }
}
```

Electronic Source Code:

```
import UIKit
struct Electronic {
    let title: String
    let price: String
    let image: UIImage
}
let electronics: [Electronic] = [
    Electronic(title: "Baseus Power Bank 10000mAh Mini Magnetic Wireless Fast Charge", price: "$42.90", image: imageLiteral(resourceName: "Baseus Power Bank 10000mAh Mini Magnetic Wireless Fast Charge ($42.90)")),
    Electronic(title: "HQAi G01 Bluetooth Earphones Hanging Neck In-ear Bluetooth Headset", price: "$2.98", image: imageLiteral(resourceName: "HQAi G01 Bluetooth Earphones Hanging Neck In-ear Bluetooth Headset ($2.98)")),
```

```
    Electronic(title: "JISULIFE Portable Fan Mini Fan", price: "$11.90", image:  imageLiteral(resourceName: "JISULIFE Portable Fan Mini Fan ($11.90)")),

    Electronic(title: "P9 Headphone Wireless Bluetoset Sports Headphone", price: "$5.34", image: imageLiteral(resourceName: "P9 Headphone Wireless Bluetoset Sports Headphone ($5.34)")),

    Electronic(title: "Razer Viper Ultimate Wireless Gaming Mouse", price: "$145.76", image:  imageLiteral(resourceName: "Razer Viper Ultimate Wireless Gaming Mouse ($145.76)")),

    Electronic(title: "Remote Control E27 Fanco Ceiling Fan With Light Small", price: "$19.90", image: imageLiteral(resourceName: "Remote Control E27 Fanco Ceiling Fan With Light Small ($19.90)")),

    Electronic(title: "Tune 230NC TWS WAVE300 Noise Cancelling Earbuds", price: "$$108.99", image: imageLiteral(resourceName: "Tune 230NC TWS WAVE300 Noise Cancelling Earbuds ($$108.99)")),

    Electronic(title: "USB Wired Backlight Gaming Keyboard", price: "$25.38", image:  imageLiteral(resourceName: "USB Wired Backlight Gaming Keyboard ($25.38)")),

]
```

ElectronicCollectionViewCell Class Source Code:

```
//
//  ElectronicCollectionViewCell.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 12/9/24.
//
import UIKit
class ElectronicCollectionViewCell: UICollectionViewCell {
    @IBOutlet weak var electronicImageView: UIImageView!
    @IBOutlet weak var electronicTitleLabel: UILabel!
    @IBOutlet weak var electronicPriceLabel: UILabel!

    func setup(with electronic: Electronic) {
        electronicImageView.image = electronic.image
        electronicTitleLabel.text = electronic.title
        electronicPriceLabel.text = electronic.price
    }
}
```

ElectronicViewController Source Code

```
//
```

```
//  ElectronicViewController.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 12/9/24.
//
import UIKit
// used to create audio video players and play
import AVFoundation
// used for create AV Player View -- acceleration audio, forward...
import AVKit
class ElectronicViewController: UIViewController {
    @IBOutlet weak var collectionView: UICollectionView!
    let file_vid_url = Bundle.main.url(forResource: "ShopeeAdvert2019", withExtension: "mov")
    // Video player object
    var Vidplayer: AVPlayer!
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        // Initialize the video player
        if let path = file_vid_url
        {
            // Try to create a video player with the provided URL
            Vidplayer = try! AVPlayer(url: path)
        }
        else
        {
            // Print an error message if the video file is not found
            print("Video file not found")
        }
        collectionView.dataSource = self
        collectionView.delegate = self
        collectionView.collectionViewLayout = UICollectionViewFlowLayout()
    }
```

```
    @IBAction func playElectronicVideo(_ sender: Any) {
        // Create an AVPlayerViewController to display the video
        let AVcontroller = AVPlayerViewController()
        // Set the video player for the AVPlayerViewController
        AVcontroller.player = Vidplayer
        // Present the AVPlayerViewController and start playing the video
        present(AVcontroller, animated: true)
        {
            AVcontroller.player!.play()
        }
    }
}


extension ElectronicViewController: UICollectionViewDataSource {
    func collectionView(_ collectionView: UICollectionView, numberOfItemsInSection section: Int) -> Int {
        return electronics.count
    }


    func collectionView(_ collectionView: UICollectionView, cellForItemAt indexPath: IndexPath) -> UICollectionViewCell
{
        let cell = collectionView.dequeueReusableCell(withReuseIdentifier: "ElectronicCollectionViewCell", for: indexPath)
as! ElectronicCollectionViewCell
        cell.setup(with: electronics[indexPath.row])
        return cell
    }
}


extension ElectronicViewController: UICollectionViewDelegateFlowLayout {
    func collectionView(_ collectionView: UICollectionView, layout collectionViewLayout: UICollectionViewLayout,
sizeForItemAt indexPath: IndexPath) -> CGSize {
        return CGSize(width: 200, height: 300)
    }
}


extension ElectronicViewController: UICollectionViewDelegate {
```

```
    func collectionView(_ collectionView: UICollectionView, didSelectItemAt indexPath: IndexPath) {
        print(electronics[indexPath.row].title)
    }
}
```

MapViewController Source Code:

```
//
//  MapViewController.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 13/9/24.
//
import UIKit
import MapKit
import CoreLocation
class MapViewController: UIViewController, MKMapViewDelegate
{
    @IBOutlet weak var map: MKMapView!

    // Singapore Coordinates
    let SG_coordinate = CLLocationCoordinate2D(
        latitude: 1.290270,
        longitude: 103.851959
    )
    // SUSS Coordinates
    let SUSS_coordinate = CLLocationCoordinate2D(
        latitude: 1.3291,
        longitude: 103.7762
    )
    // SUSS Coordinates
    let Sentosa_coordinate = CLLocationCoordinate2D(
        latitude: 1.2494,
        longitude: 103.8303
```

```
)

override func viewDidLoad() {
    super.viewDidLoad()
    // Do any additional setup after loading the view.
    view.addSubview(map)
    map.frame = view.bounds
    map.setRegion(MKCoordinateRegion(
        center: SG_coordinate,
        span: MKCoordinateSpan(
            latitudeDelta: 0.1,
            longitudeDelta: 0.1)
    ), animated: false)
    map.delegate = self
    addCustomPin()
    // Call the new function to add the SUSS pin
    addSUSSPokeballPin()
    // Call the new function to add the Sentosa pin
    addSentosaPokeballPin()
}
private func addCustomPin()
{
    let pin = MKPointAnnotation()
    pin.coordinate = SG_coordinate
    pin.title = "Singapore"
    pin.subtitle = "Pokemon Here in Singapore"
    map.addAnnotation(pin)
}
// Add a new function to add the SUSS pin
private func addSUSSPokeballPin() {
    let sussPin = MKPointAnnotation()
    sussPin.coordinate = SUSS_coordinate
    sussPin.title = "SUSS"
    sussPin.subtitle = "Catch 'em all at SUSS!"
```

```
        map.addAnnotation(sussPin)
    }
    // Add a new function to add the SUSS pin
    private func addSentosaPokeballPin() {
        let sentosaPin = MKPointAnnotation()
        sentosaPin.coordinate = Sentosa_coordinate
        sentosaPin.title = "Sentosa"
        sentosaPin.subtitle = "Catch 'em all at Sentosa!"
        map.addAnnotation(sentosaPin)
    }
    // Map
    func mapView(_ mapView: MKMapView, viewFor annotation: MKAnnotation) -> MKAnnotationView?
    {
        guard !(annotation is MKUserLocation) else {
            return nil
        }
        var annotationView = map.dequeueReusableAnnotationView(withIdentifier: "custom")
        if annotationView == nil {
            // Create the view
            annotationView = MKAnnotationView(
                annotation: annotation,
                reuseIdentifier: "custom"
            )
            annotationView?.canShowCallout = true
        }
        else {
            annotationView?.annotation = annotation
        }
        annotationView?.image = UIImage(named: "Pokeball_4")
        return annotationView
    }
}
```

CameraPhotosViewController Source Code:

```
//
//  CameraPhootsViewController.swift
//  The Quicky Mart
//
//  Created by Shawn Yang on 13/9/24.
//
import UIKit
class CameraPhotosViewController: UIViewController {
    @IBOutlet weak var imageFrame: UIImageView!
    @IBOutlet weak var cameraButton: UIButton!
    @IBOutlet weak var photoGalleryButton: UIButton!

    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        imageFrame.backgroundColor = .secondarySystemBackground
        cameraButton.backgroundColor = .systemBlue
        cameraButton.setTitle("Camera", for: .normal)
        cameraButton.setTitleColor(.white, for: .normal)
    }

    @IBAction func didTapButton ()
    {
        let picker = UIImagePickerController()
        picker.sourceType = .camera
        picker.allowsEditing = true
        picker.delegate = self
        present(picker, animated: true)
    }

    @IBAction func photoGalleryDidTapButton ()
    {
        let vc = UIImagePickerController()
```

```
        vc.sourceType = .photoLibrary
        vc.delegate = self
        vc.allowsEditing = true
        present(vc, animated: true)
    }
}

extension CameraPhotosViewController: UIImagePickerControllerDelegate, UINavigationControllerDelegate
{
    func imagePickerControllerDidCancel(_ picker: UIImagePickerController)
    {
        picker.dismiss(animated: true, completion: nil)
    }

    func imagePickerController(_ picker: UIImagePickerController, didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any])
    {
//        print("\(info)")
        picker.dismiss(animated: true, completion: nil)
        guard let image = info[UIImagePickerController.InfoKey.editedImage] as? UIImage else {
            return
        }
        imageFrame.image = image
        if let image_2 = info[UIImagePickerController.InfoKey(rawValue: "UIImagePickerControllerEditedImage")] as? UIImage {
            imageFrame.image = image_2
        }
    }
}
```

GesturesViewController Source Code:

```
//
//  GesturesViewController.swift
//  The Quicky Mart
```

```
//
//  Created by Shawn Yang on 13/9/24.
//
import UIKit
class GesturesViewController: UIViewController {
    @IBOutlet weak var indigoView: UIView!
    var offset: CGPoint?
    override func viewDidLoad() {
        super.viewDidLoad()
        // Do any additional setup after loading the view.
        let myView = UIView(frame: CGRect(x: 0, y: 0, width: 200, height: 200))
        myView.backgroundColor = .red
        myView.center = view.center
        view.addSubview(myView)
        let gestureRecognizer = UITapGestureRecognizer(target: self, action: #selector(gestureFired(_:)))
        gestureRecognizer.numberOfTapsRequired = 2
        gestureRecognizer.numberOfTouchesRequired = 1
        myView.addGestureRecognizer(gestureRecognizer)
        myView.isUserInteractionEnabled = true
        let gestureRecognizer_2 = UISwipeGestureRecognizer(target: self, action: #selector(gestureFired_2(_:)))
        gestureRecognizer_2.direction = .right
        gestureRecognizer_2.numberOfTouchesRequired = 1
        myView.addGestureRecognizer(gestureRecognizer_2)
        myView.isUserInteractionEnabled = true
        let panGesture = UIPanGestureRecognizer(target: self, action: #selector(panned))
        indigoView.addGestureRecognizer(panGesture)
        panGesture.delegate = self
    }

    @objc func panned(_ gesture: UIPanGestureRecognizer) {
        print("panned")
        let translation = gesture.translation(in: indigoView)
        indigoView.frame.origin.x += translation.x
        indigoView.frame.origin.y += translation.y
```

```
        gesture.setTranslation(.zero, in: indigoView)
    }

    @objc func gestureFired(_ gesture: UITapGestureRecognizer)
    {
        if let fireView = gesture.view {
            fireView.backgroundColor = .blue
        }
    }

    @objc func gestureFired_2(_ gesture: UISwipeGestureRecognizer) {
        if let fireView = gesture.view {
            fireView.backgroundColor = .green
        }
    }

    @IBAction func longPressed(_ sender: UILongPressGestureRecognizer)
    {
        if sender.state == .began {
            indigoView.transform = CGAffineTransform(scaleX: 1.25, y: 1.25)
        }
        if sender.state == .ended {
            indigoView.transform = .identity
        }
    }
}
extension GesturesViewController: UIGestureRecognizerDelegate {
    func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer, shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer) -> Bool
    {
        return true
    }
}
```

---- END OF ASSIGNMENT ----