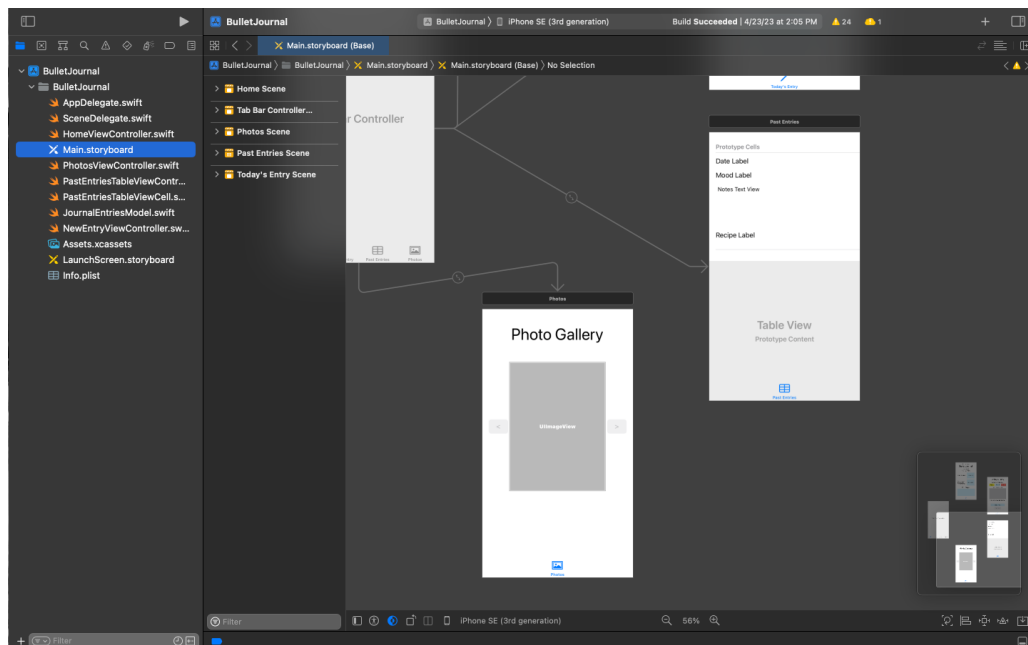
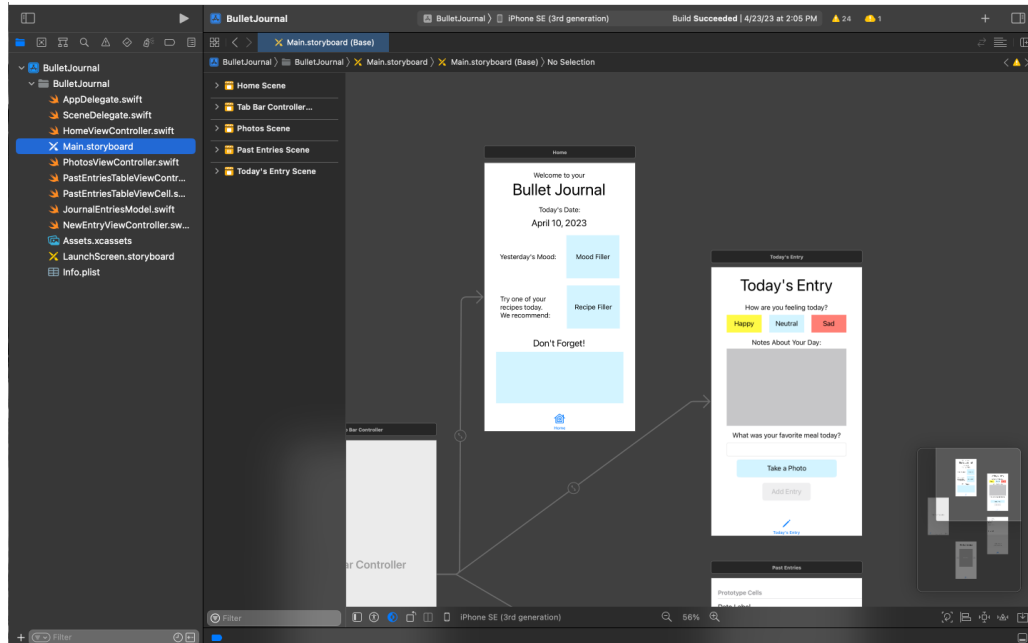
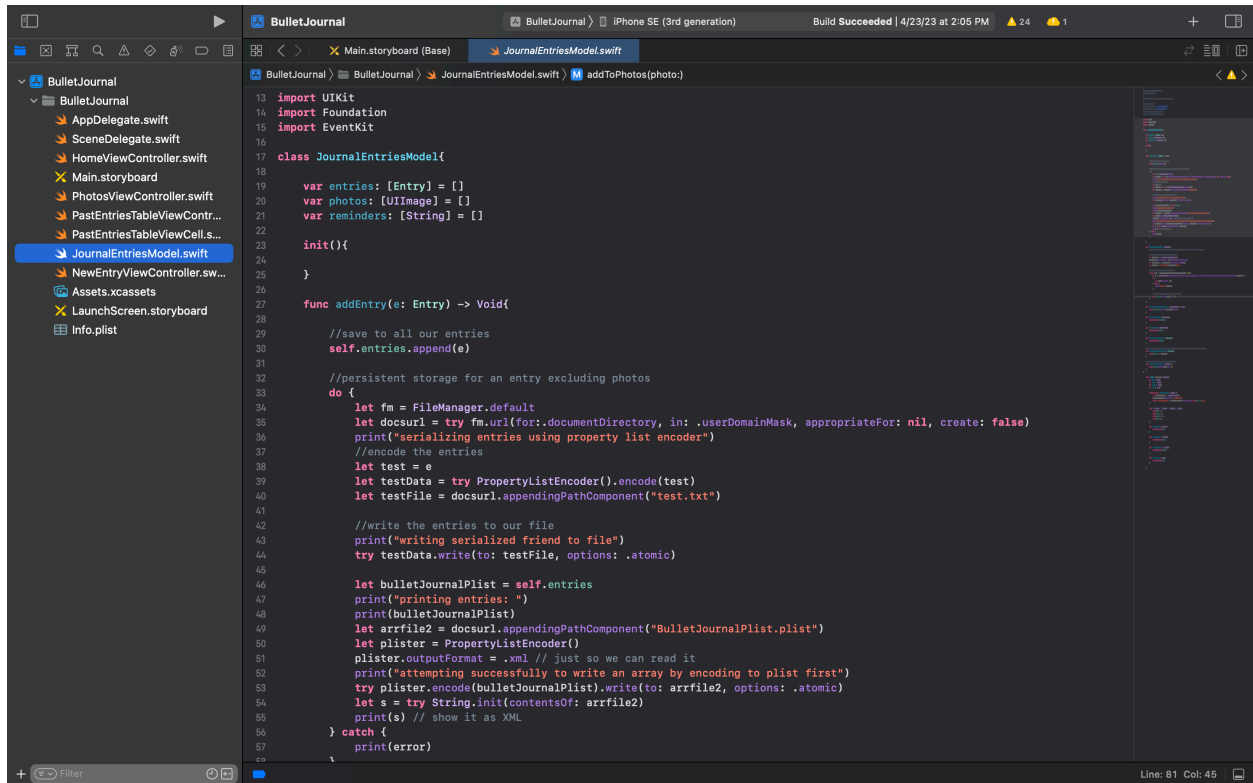


Bullet Journal App

The UI

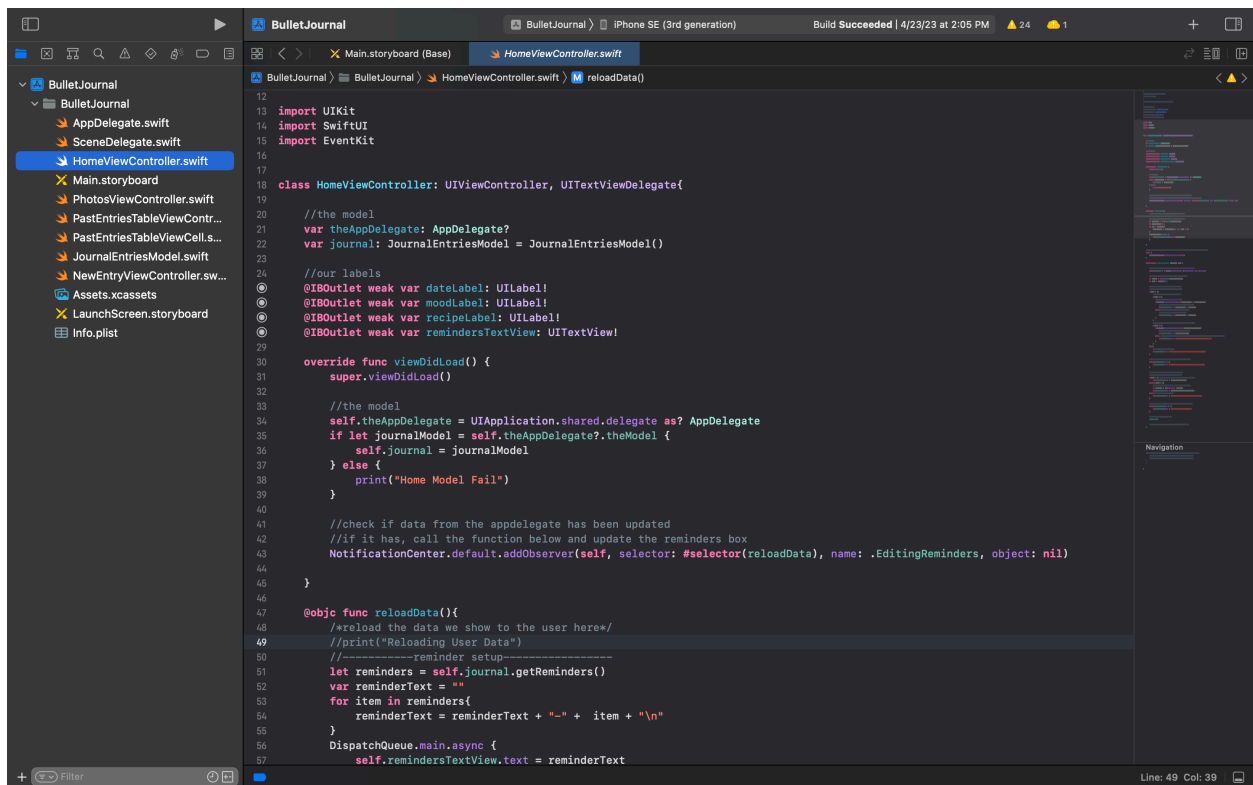


A Glimpse at the Code



The screenshot shows the Xcode IDE with the 'BulletJournal' project open. The left sidebar displays the project structure, with 'JournalEntriesModel.swift' selected. The main editor area shows the Swift code for this model. The code includes imports for UIKit, Foundation, and EventKit. It defines a 'JournalEntriesModel' class with three arrays: 'entries' of type 'Entry', 'photos' of type 'UIImage', and 'reminders' of type 'String'. The 'init()' method initializes these arrays. The 'addEntry(e: Entry) -> Void' method is shown, which appends the entry to the 'entries' array and then saves the data to a file named 'BulletJournalPlist.plist' using PropertyListEncoder. The status bar at the bottom indicates 'Line: 81 Col: 45'.

```
13 import UIKit
14 import Foundation
15 import EventKit
16
17 class JournalEntriesModel{
18
19     var entries: [Entry] = []
20     var photos: [UIImage] = []
21     var reminders: [String] = []
22
23     init(){
24
25     }
26
27     func addEntry(e: Entry) -> Void{
28
29         //save to all our entries
30         self.entries.append(e)
31
32         //persistent storage for an entry excluding photos
33         do {
34             let fm = FileManager.default
35             let docurl = try fm.url(for:.documentDirectory, in: .userDomainMask, appropriateFor: nil, create: false)
36             print("serializing entries using property list encoder")
37             //encode the entries
38             let test = e
39             let testData = try PropertyListEncoder().encode(test)
40             let testFile = docurl.appendingPathComponent("test.txt")
41
42             //write the entries to our file
43             print("writing serialized friend to file")
44             try testData.write(to: testFile, options: .atomic)
45
46             let bulletJournalPlist = self.entries
47             print("printing entries: ")
48             print(bulletJournalPlist)
49             let arrfile2 = docurl.appendingPathComponent("BulletJournalPlist.plist")
50             let plister = PropertyListEncoder()
51             plister.outputFormat = .xml // just so we can read it
52             print("attempting successfully to write an array by encoding to plist first")
53             try plister.encode(bulletJournalPlist).write(to: arrfile2, options: .atomic)
54             let s = try String.init(contentsOf: arrfile2)
55             print(s) // show it as XML
56         } catch {
57             print(error)
58         }
59     }
60 }
```



The screenshot shows the Xcode IDE with the 'BulletJournal' project open. The left sidebar displays the project structure, with 'HomeController.swift' selected. The main editor area shows the Swift code for this controller. The code includes imports for UIKit, SwiftUI, and EventKit. It defines a 'HomeController' class that inherits from 'UIViewController' and 'UITextViewDelegate'. The 'viewDidLoad()' method is overridden to initialize the app delegate and the journal model. The 'reloadData()' method is implemented to reload the data from the app delegate and update the reminders text view. The status bar at the bottom indicates 'Line: 49 Col: 39'.

```
12
13 import UIKit
14 import SwiftUI
15 import EventKit
16
17 class HomeController: UIViewController, UITextViewDelegate{
18
19     //the model
20     var appDelegate: AppDelegate?
21     var journal: JournalEntriesModel = JournalEntriesModel()
22
23     //our labels
24     @IBOutlet weak var dateLabel: UILabel!
25     @IBOutlet weak var moodLabel: UILabel!
26     @IBOutlet weak var recipeLabel: UILabel!
27     @IBOutlet weak var remindersTextView: UITextView!
28
29     override func viewDidLoad() {
30         super.viewDidLoad()
31
32         //the model
33         self.appDelegate = UIApplication.shared.delegate as? AppDelegate
34         if let journalModel = self.appDelegate?.theModel {
35             self.journal = journalModel
36         } else {
37             print("Home Model Fail")
38         }
39     }
40
41     //check if data from the appdelegate has been updated
42     //if it has, call the function below and update the reminders box
43     NotificationCenter.default.addObserver(self, selector: #selector(reloadData), name: .EditingReminders, object: nil)
44
45     @objc func reloadData(){
46         //reload the data we show to the user here*/
47         //print("Reloading User Data")
48         //-----reminder setup-----
49         let reminders = self.journal.getReminders()
50         var reminderText = ""
51         for item in reminders{
52             reminderText = reminderText + "-" + item + "\n"
53         }
54         DispatchQueue.main.async {
55             self.remindersTextView.text = reminderText
56         }
57     }
58 }
```