

TO-DO APPLICATION FOR IOS

A MINOR PROJECT REPORT

Submitted by

**SRI ABIRAM G B [RA2011004010211]
KANISHK K U[RA2011004010226]
RAMKUMAR M V [RA2011004010079]**

Under the guidance of

Dr.N.Arunachalam
(Assistant Professor)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

of

COLLEGE OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Chengalpattu District

NOVEMBER 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

ABSTRACT

One of the most important reasons you should use a to do list is that it will help you stay organized. When you write all your tasks in a list, they seem more manageable. When you've got a clear outline of the tasks you've got to do and those you've completed, it helps you stay focused. While freeing up space in your mind for other more creative tasks. When you complete a task, you can cross it off your list. This gives you a sense of progress and achievement, something you'll lack if you're always rushing from one task to the next. If you feel a sense of achievement, it spurs you on and motivates you to keep moving forward.

Introduction:

It's sometimes hard to know which of the many items in your to-do list you should do now. This could be because you have other events coming up you need to attend, and there isn't time to do the most important things. Instead, you might find yourself wasting time or doing an unimportant smaller task, when there is something more important you could have completed if you knew about it.

Certain tasks require more concentration and effort than others. If at any given time you do not feel up to completing a task that requires significant focus, you should do the most important thing that requires the least effort. This type of prioritizing isn't often managed by todo lists or calendars, and that's one of the major changes we hoped to have in ours.

Our goal was to create a to-do list that rearranges itself around a calendar based on the due date and difficulty of each task. In addition, we wanted to familiarize ourselves with Android APIs, to rethink the role of the calendar and how mobile devices represent calendars, and to implement a basic auto-scheduling tool. There are existing to-do list applications and calendar applications, but nothing that attempts to integrate the two in a useful manner. We decided that the best way to achieve our goals in the context of Software Engineering was to create an Android application.

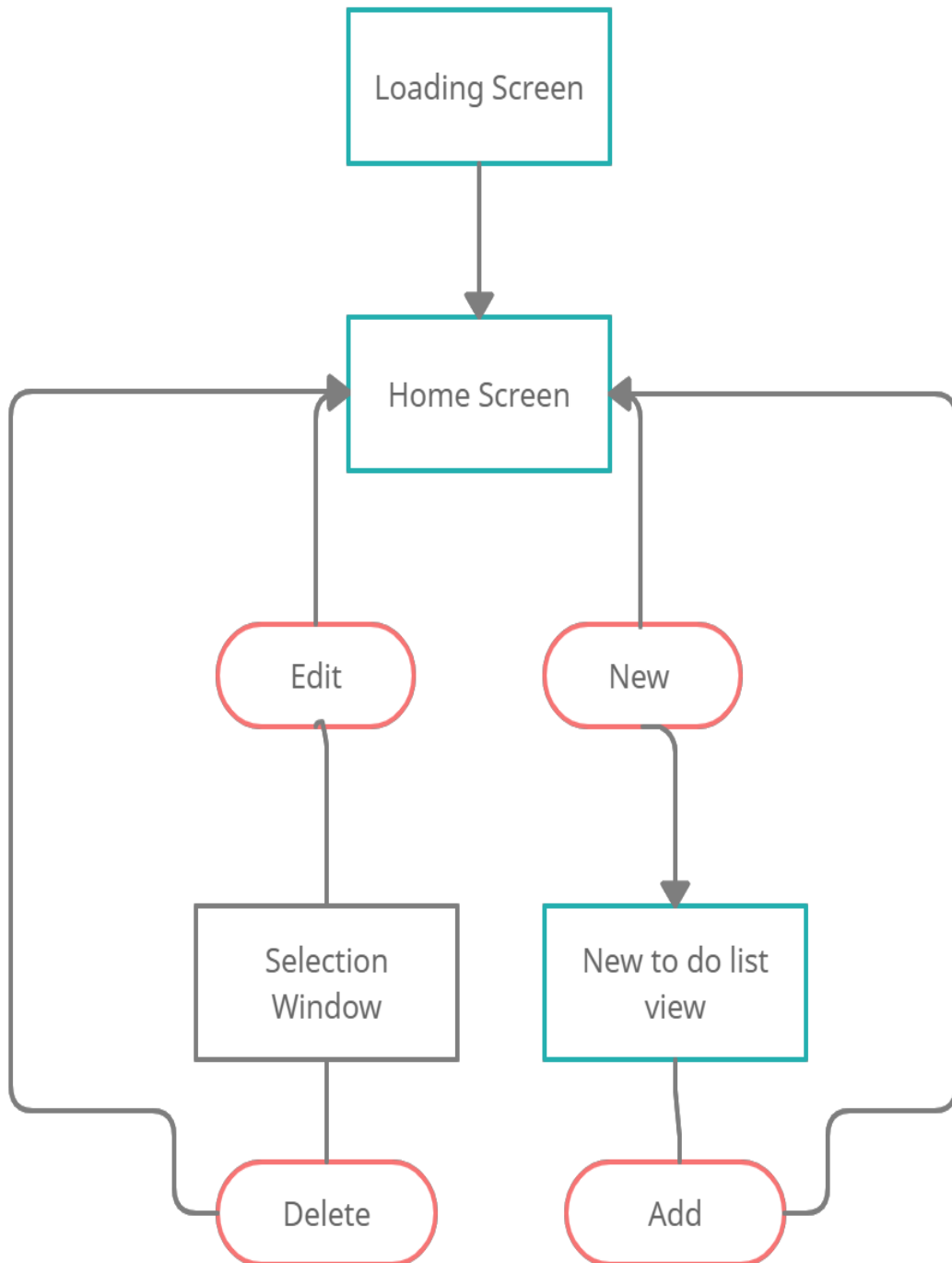
We chose a paradigm where the application determines importance from user entry of due date, difficulty level, and amount of time required. The app returns the list with the to-dos in order of due date and orders the to-dos for each date from hardest to easiest. The app also pulls events from the local Android calendar. Based on available blocks of time, it prioritizes the tasks to fit the available time as shown in the following images.

Features:

The basic elements of a to-do list will depend on the overall tasks it contains. For the most part, these elements are:

- Tasks organized from most to least importance
- Items centered around work priorities, personal projects, or a combination
- Deadlines for list items
- The right software to keep all list items organized, which can be either be a to-do list app, or task management or project management tools
- Items that are as specific as possible
- Make the list easily accessible so it can be checked and tracked each day

Block Diagram



Coding

```
import Foundation
```

```
final class ListViewModel:
```

```
ObservableObject {
```

```
    //MARK: - Variables
```

```
    @Published var items: [ItemModel] =
```

```
    [] {
```

```
        didSet {
```

```
            saveItems()
```

```
        }
```

```
    }
```

```
    private let itemsKey: String =
```

```
"items_list"
```

```
    //MARK: - Init
```

```
    init() {
```

```
        getItems()
```

```
    }
```

```
    //MARK: - Methods
```

```
    private func getItems() {
```

```
        guard let data =
```

```
UserDefaults.standard.data(forKey:
```

```
itemsKey) else { return }
```

```
        do {
```

```
            items = try
```

```
JSONDecoder().decode([ItemModel].self,
```

```
from: data)
```

```
        } catch {
```

```
            print(error.localizedDescription)
```

```
        }
```

```
    }
```

```

func removeItem(at index: IndexSet) {
    items.remove(atOffsets: index)
}

func moveItem(from: IndexSet, to: Int)
{
    items.move(fromOffsets: from,
toOffset: to)
}

func addItem(title: String) {
    let newItem = ItemModel(title: title,
isCompleted: false)
    items.append(newItem)
}

func updateItem(item: ItemModel) {
    if let index = items.firstIndex(where:
{ $0.id == item.id }) {
        items[index] =
item.updateCompletion()
    }
}

func saveItems() {
    if let encodedData = try?
JSONEncoder().encode(items) {

UserDefaults.standard.setValue(encoded
Data, forKey: itemsKey)
    }
}
}

```

```

import SwiftUI

struct ListView: View {
    //MARK: - Variables
    @EnvironmentObject var
    listViewModel: ListViewModel

    //MARK: - Body
    var body: some View {
        ZStack {
            if listViewModel.items.isEmpty {
                NoItemsView()

                .transition(AnyTransition.opacity.animation(
                    on(.easeIn))
                } else {
                    List {

                        ForEach(listViewModel.items) { item in
                            ListRowView(item: item)
                                .onTapGesture {

                                    withAnimation(.linear) {


                                        listViewModel.updateItem(item: item)
                                            }
                                            }
                                    }
                                    .onDelete(perform:
listViewModel.removeItem(at:))
                                    .onMove(perform:
listViewModel.moveItem(from:to:))
                                            }
                                            .listStyle(.plain)
                                    }

```



```

    }

    .navigationTitle("Todo List )

    .navigationBarItems(
        leading: EditButton(),
        trailing: NavigationLink("Add",
destination: AddView())
    )
}
}

```

```

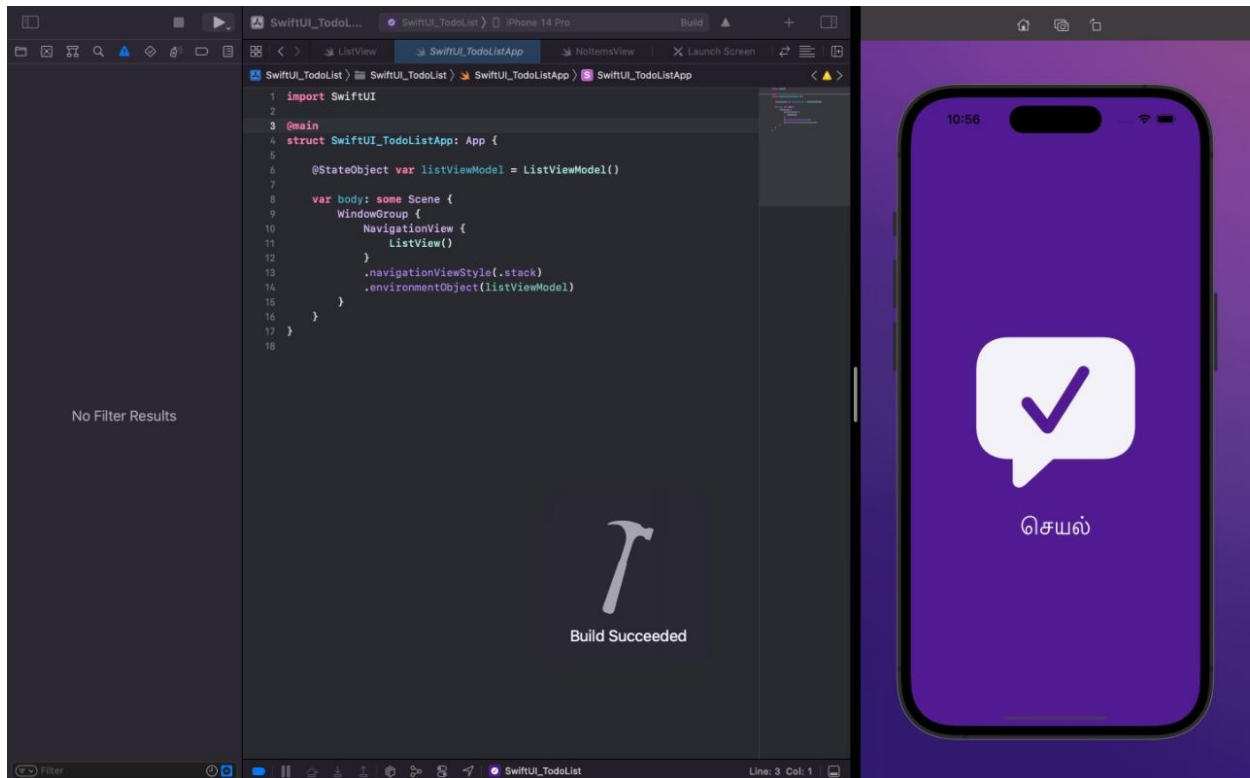
struct ListView_Previews:
PreviewProvider {
    static var previews: some View {
        NavigationView {
            ListView()
        }
    }

    .environmentObject(ListViewModel())
}
}

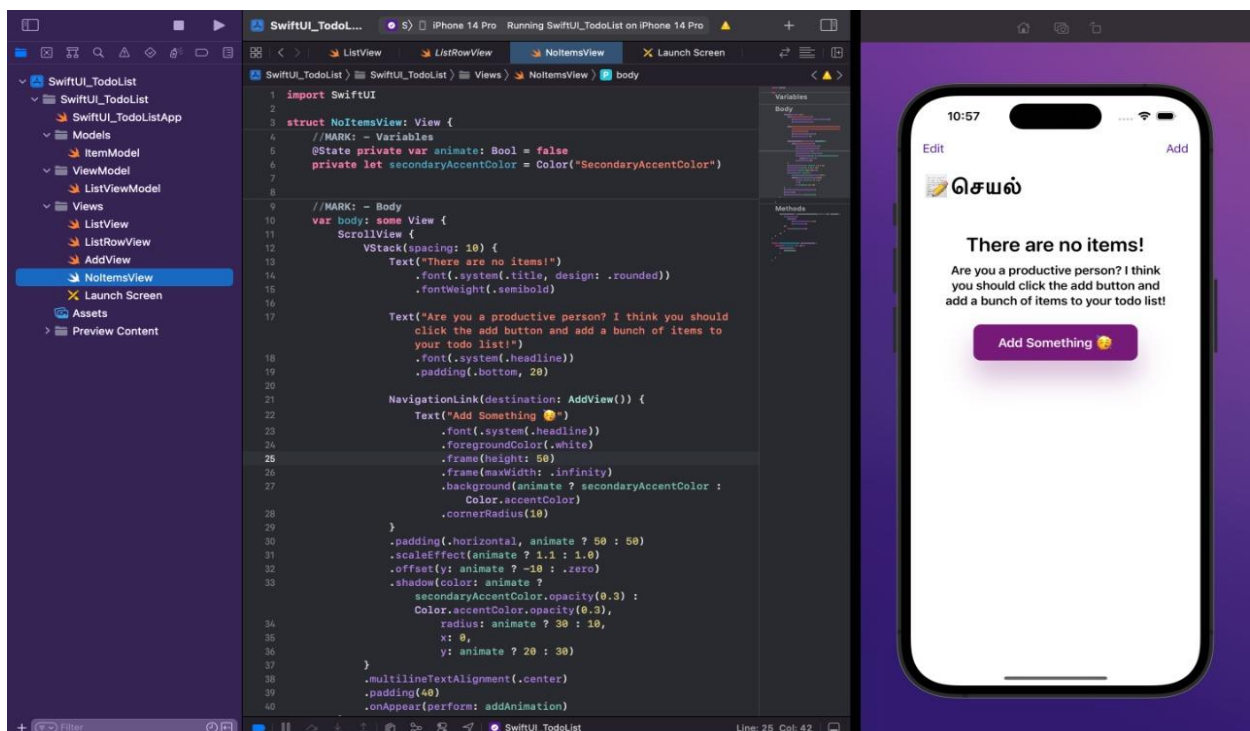
```

Views

Loading screen View



Home Screen View (with no activities to do)



```

import SwiftUI

struct NoItemsView: View {
    //MARK: - Variables
    @State private var animate: Bool =
false
    private let secondaryAccentColor =
Color("SecondaryAccentColor")

    //MARK: - Body
    var body: some View {
        ScrollView {
            VStack(spacing: 10) {
                Text("There are no items!")
                    .font(.system(.title, design:
.rounded))
                    .fontWeight(.semibold)

                Text("Are you a productive
person? I think you should click the add
button and add a bunch of items to your
todo list!")
                    .font(.system(.headline))
                    .padding(.bottom, 20)

                NavigationLink(destination:
AddView()) {
                    Text("Add Something 🎉")
                        .font(.system(.headline))
                        .foregroundColor(.white)
                        .frame(height: 55)
                        .frame(maxWidth:
.infinity)
                }
            }
        }
    }
}

```

```

        .background(animate ?
secondaryAccentColor :
Color.accentColor)
        .cornerRadius(10)
    }
    .padding(.horizontal, animate ?
30 : 50)
    .scaleEffect(animate ? 1.1 : 1.0)
    .offset(y: animate ? -10 : .zero)
    .shadow(color: animate ?
secondaryAccentColor.opacity(0.7) :
Color.accentColor.opacity(0.7),
        radius: animate ? 30 : 10,
        x: 0,
        y: animate ? 50 : 30)
    }
    .multilineTextAlignment(.center)
    .padding(40)
    .onAppear(perform:
addAnimation)
    }

}

```

//MARK: - Methods

```

private func addAnimation() {
    guard !animate else { return }

```

```

DispatchQueue.main.asyncAfter(deadline
: .now() + 1.5, execute: {
    withAnimation(
        Animation
        .easeInOut(duration: 2.0)
        .repeatForever()
    ) {

```

```

        self.animate.toggle()
    }
    })
}
}

```

```

struct NoItemsView_Previews:

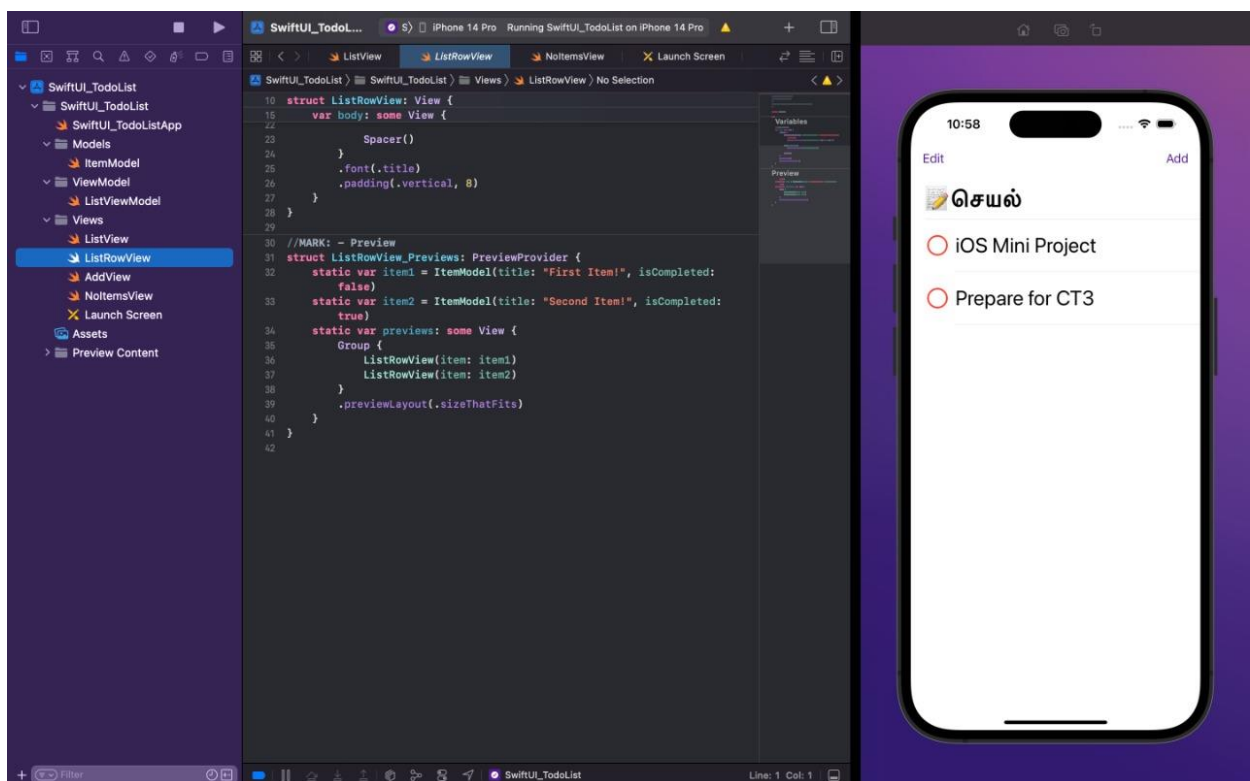
```

```

    PreviewProvider {
        static var previews: some View {
            NavigationView {
                NoItemsView()
            }
        }
    }
}

```

Home Screen View (with to do items)



```

import SwiftUI

```

```

struct ListView: View {
    //MARK: - Variables

```

```


@EnvironmentObject var
listViewModel: ListViewModel

//MARK: - Body
var body: some View {
    ZStack {
        if listViewModel.items.isEmpty {
            NoItemsView()

.transition(AnyTransition.opacity.animation(
on(.easeIn))
        } else {
            List {

ForEach(listViewModel.items) { item in
                ListRowView(item: item)
                    .onTapGesture {

withAnimation(.linear) {

listViewModel.updateItem(item: item)
                    }
                }
            }
            .onDelete(perform:
listViewModel.removeItem(at:))
            .onMove(perform:
listViewModel.moveItem(from:to:))
                }
            .listStyle(.plain)
        }
    }
    .navigationTitle("Todo List )
    .navigationBarItems(

```

```

        leading: EditButton(),
        trailing: NavigationLink("Add",
destination: AddView())
    )
}
}

```

struct ListView_Previews:

```

PreviewProvider {
    static var previews: some View {
        NavigationView {
            ListView()
        }
    }
}

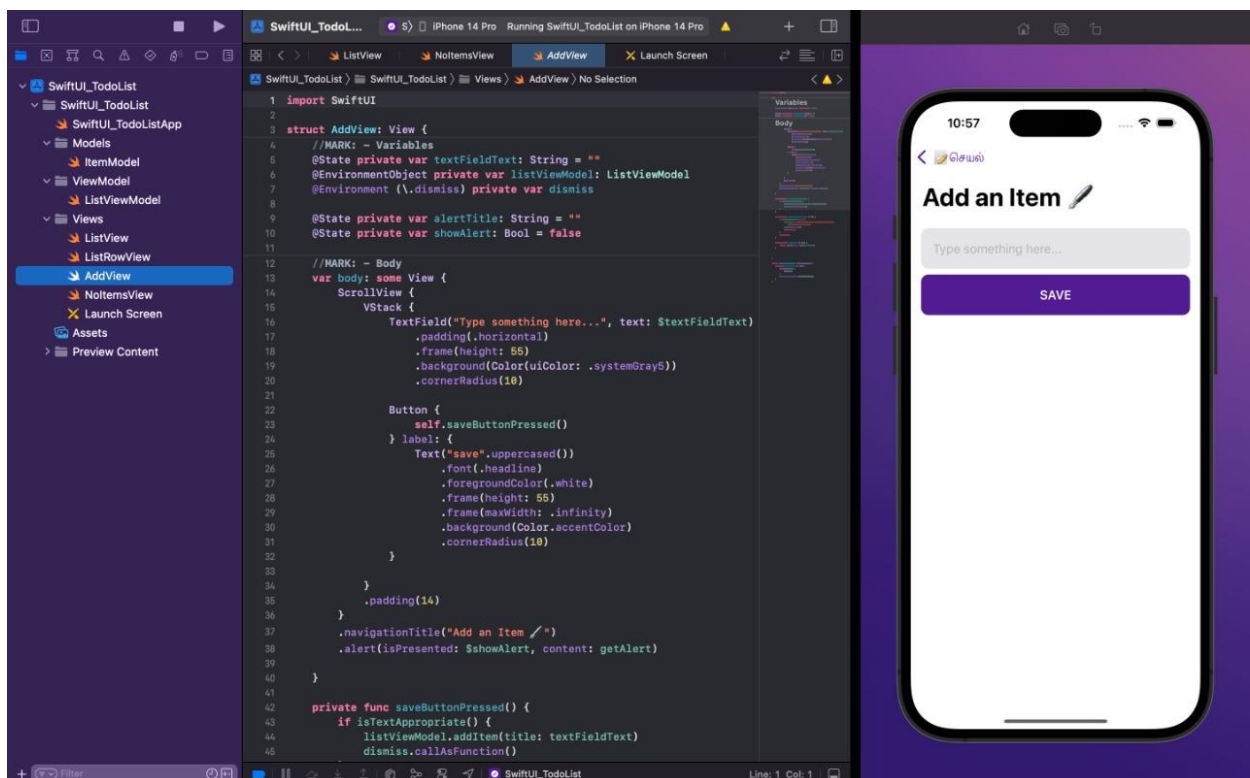
```

```

.environmentObject(ListViewModel())
}
}

```

Adding View



```

import SwiftUI

struct AddView: View {
    //MARK: - Variables
    @State private var textFieldText:
String = ""
    @EnvironmentObject private var
listViewModel: ListViewModel
    @Environment (\.dismiss) private var
dismiss

    @State private var alertTitle: String =
""
    @State private var showAlert: Bool =
false

    //MARK: - Body
    var body: some View {
        ScrollView {
            VStack {
                TextField("Type something
here...", text: $textFieldText)
                    .padding(.horizontal)
                    .frame(height: 55)
                    .background(Color(uiColor:
.systemGray5))
                    .cornerRadius(10)

                Button {
                    self.saveButtonPressed()
                } label: {
                    Text("save".uppercased())
                        .font(.headline)
                        .foregroundColor(.white)
                        .frame(height: 55)

```



```

        .frame(maxWidth:
.infinity)

        .background(Color.accentColor)
            .cornerRadius(10)
        }

    }

    .padding(14)
}

.navigationTitle("Add an Item ✍️")
.alert(isPresented: $showAlert,
content: getAlert)

}

```

```

private func saveButtonPressed() {
    if isTextAppropriate() {
        listViewModel.addItem(title:
textFieldText)
        dismiss.callAsFunction()
    }
}

```

```

private func isTextAppropriate() ->
Bool {
    if textFieldText.count < 3 {
        alertTitle = "Your new todo item
must be at least 3 characters long!!
😬😬"
        showAlert.toggle()
        return false
    }
    return true
}

```

```
}
```

```
private func getAlert() -> Alert {  
    return Alert(title: Text(alertTitle))  
}
```

```
}
```

```
struct AddView_Previews:
```

```
PreviewProvider {  
    static var previews: some View {  
        NavigationView {  
            AddView()  
        }  
    }  
}
```

```
.environmentObject(ListViewModel())  
}  
}
```

Conclusion:

It was a great experience in the field of IOS development and to make a much needed application that makes us productive.

Reference:

https://github.com/Ahmed-Amin-Hassan-Ismail/SwiftUI_TODO_APP