//Logic.Swift

```
import SwiftUI
// Simple model for a task with basic properties
struct Task: Identifiable {
    var id = UUID()
    var title: String
    var isCompleted: Bool
    var dueDate: Date
}
// This class manages our collection of tasks
class TaskManager: ObservableObject {
    // This is our list collection that stores all tasks
    @Published var tasks: [Task] = []

    // Add a new task to our collection
    func addTask(title: String, dueDate: Date) {
        let newTask = Task(id: UUID(), title: title, isCompleted: false, dueDate: dueDate)
        tasks.append(newTask)
    }

    // Toggle completion status of a task
    func toggleCompletion(at index: Int) {
        if index >= 0 && index < tasks.count {
            tasks[index].isCompleted.toggle()
        }
    }

    // Remove a task from our collection
    func deleteTask(at index: Int) {
        if index >= 0 && index < tasks.count {
            tasks.remove(at: index)
        }
    }

    // This procedure filters tasks by a search term
    // Parameters: keyword - the search term to look for
    // Returns: a filtered list of tasks matching the keyword
    func searchTasks(keyword: String) -> [Task] {
        // Create empty result list
```

```
        var results: [Task] = []

        // Iterate through all tasks to find matches
        for task in tasks {
            // Selection: Check if task title contains the keyword
            if task.title.lowercased().contains(keyword.lowercased()) {
                // Add matching task to results
                results.append(task)
            }
        }

        // Return the filtered collection
        return results
    }
}
// Main app entry point
@main
struct DayPlannerApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

//ContentView.Swift

```
// Acknowledgment:During the development of this program, my brother assisted by providing feedback and suggestions for the user interface design and helped troubleshoot some syntax errors when I got stuck. All algorithms, data structures, and core functionalities were independently designed and implemented by me.
import SwiftUI

struct ContentView: View {
    // Connect to our task manager
    @StateObject var taskManager = TaskManager()

    // UI state variables
    @State var showingAddTask = false
    @State var selectedDate = Date()
```

```
    @State var newTaskTitle = ""
    @State var newTaskDate = Date()
    @State var searchText = ""
    @State var isSearching = false

    var body: some View {
        VStack {
            // App title
            Text("Simple Day Planner")
                .font(.largeTitle)
                .padding()

            // User input: Date selection
            DatePicker("Select Date:", selection: $selectedDate, displayedComponents:
.date)
                .padding()

            // User input: Search box (only shown when searching)
            if isSearching {
                HStack {
                    TextField("Search tasks...", text: $searchText)
                        .textFieldStyle(RoundedBorderTextFieldStyle())
                        .padding(.horizontal)

                    Button("Cancel") {
                        isSearching = false
                        searchText = ""
                    }
                    .padding(.trailing)
                }
            }

            // Output: Display of tasks based on input
            List {
                // Determine which tasks to show (searched or date-filtered)
                let displayedTasks = isSearching ?
                    taskManager.searchTasks(keyword: searchText) :
                    tasksForSelectedDate()

                if displayedTasks.isEmpty {
```

```
                Text("No tasks found")
                    .foregroundColor(.gray)
                    .italic()
            }

            ForEach(displayedTasks) { task in
                // Task display row
                HStack {
                    // Checkbox button for completion
                    Button(action: {
                        // Find index in the main task list
                        if let index = taskManager.tasks.firstIndex(where: { $0.id == task.id }) {
                            // Call our procedure to toggle completion
                            taskManager.toggleCompletion(at: index)
                        }
                    }) {
                        Image(systemName: task.isCompleted ? "checkmark.square" :
"square")
                    }

                    // Task title with strikethrough when completed
                    Text(task.title)
                        .strikethrough(task.isCompleted)
                        .foregroundColor(task.isCompleted ? .gray : .primary)
                }
            }
            .onDelete { indexSet in
                // Handle delete operations
                let filteredTasks = isSearching ?
                    taskManager.searchTasks(keyword: searchText) :
                    tasksForSelectedDate()

                for index in indexSet {
                    let task = filteredTasks[index]
                    if let realIndex = taskManager.tasks.firstIndex(where: { $0.id == task.id })
{
                        // Call our procedure to delete task
                        taskManager.deleteTask(at: realIndex)
                    }
                }
```

```
            }
        }

        HStack {
            // Button to add a new task
            Button("Add New Task") {
                showingAddTask = true
            }
            .padding()
            .background(Color.blue)
            .foregroundColor(.white)
            .cornerRadius(10)

            // Button to search tasks
            Button(isSearching ? "View Calendar" : "Search Tasks") {
                isSearching.toggle()
                if !isSearching {
                    searchText = ""
                }
            }
            .padding()
            .background(Color.green)
            .foregroundColor(.white)
            .cornerRadius(10)
        }
        .padding()
    }
    // Add task dialog
    .sheet(isPresented: $showingAddTask) {
        VStack {
            Text("Add New Task")
                .font(.headline)
                .padding()

            // User input: Task name
            TextField("Task Name", text: $newTaskTitle)
                .textFieldStyle(RoundedBorderTextFieldStyle())
                .padding()

            // User input: Task date
```

```
            DatePicker("Due Date", selection: $newTaskDate)
                .padding()

            HStack {
                Button("Cancel") {
                    showingAddTask = false
                    newTaskTitle = ""
                }
                .padding()

                Button("Save Task") {
                    // Only add if we have a title
                    if !newTaskTitle.isEmpty {
                        // Call our procedure to add a task
                        taskManager.addTask(title: newTaskTitle, dueDate: newTaskDate)
                        newTaskTitle = ""
                        showingAddTask = false
                    }
                }
                .padding()
                .background(Color.blue)
                .foregroundColor(.white)
                .cornerRadius(8)
            }
        }
        .padding()
    }
}

// This procedure filters tasks for the selected date
// Returns: a list collection of tasks for that date
func tasksForSelectedDate() -> [Task] {
    // Create calendar for date comparison
    let calendar = Calendar.current

    // Filter tasks that match the selected date
    return taskManager.tasks.filter { task in
        // Check if dates are on the same day
        calendar.isDate(task.dueDate, inSameDayAs: selectedDate)
    }
```

```
    }
}
```