IOS WITH SWIFT ASSIGNMENTS

Development Scenario 1: Personal Budget Tracker

Day 1: iOS Introduction and Setup

Task 1: Configure the Xcode development environment and create a new project for the budget tracker app.

Task 2: Familiarize with the Swift language by creating basic data models for expenses and income.

Day 2: SwiftUI Basics and UI Building

Task 1: Design the main user interface using SwiftUI to display a list of transactions.

Task 2: Implement a form to add new transactions, categorizing them as either income or expenses.

Day 3: SwiftUI Layout and Views

Task 1: Use SwiftUI's layout system to create a dashboard view that shows the user's current balance and spending trends.

Task 2: Integrate SwiftUI modifiers to customize the look and feel of the dashboard according to user preferences.

Day 4: User Interaction and Navigation

Task 1: Develop interaction logic that allows users to edit or delete transactions from the list.

Task 2: Set up navigation flows to move between the dashboard, transaction list, and transaction creation form.

Day 5: State Management and Combine Framework

Task 1: Implement state management using the Combine framework to update the user interface reactively as transactions are added or modified.

Task 2: Create publishers and subscribers that handle the loading and saving of transaction data in real-time.

Day 6: Lists, Animation, and Adaptivity

Task 1: Build dynamic lists that display transactions with animations to visually distinguish between income and expenses.

Task 2: Implement gesture handling to interact with list items, such as swipe to delete.

Day 7: Advanced SwiftUI Topics and RESTful API Integration

Task 1: Add support for adaptive UIs to ensure the app looks great on all device sizes, including Dark Mode support.

Task 2: Integrate with a RESTful API to fetch real-time currency exchange rates to adjust the values of transactions made in foreign currencies.

Submission Guidelines:

1. Ensure that each answer is clear, concise, and reflects an understanding of the core concepts.

2. Diagrams can be hand-drawn and scanned or created using any digital drawing tool.

3. Provide references for any external sources used.

Submit your work in a single PDF document by end of Module.

4. You must submit your code on gitlab by the end of next day.

Development Scenario 2: Medication Reminder App

Day 1: iOS Introduction and Setup

Task 1: Set up Xcode and review the Swift language basics, focusing on syntax and control flow.

Task 2: Start a new project for a medication reminder app, establishing the basic navigation and view structure.

Day 2: SwiftUI Basics and UI Building

Task 1: Develop the user interface components for creating medication reminders, such as name, dosage, and frequency.

Task 2: Create a SwiftUI view that lists all active medication reminders.

Day 3: SwiftUI Layout and Views

Task 1: Apply layout and views in SwiftUI to design a detailed medication info card that shows next dose time and remaining pills.

Task 2: Use view modifiers to style the medication info cards and make them user-friendly.

Day 4: User Interaction and Navigation

Task 1: Handle user interactions for setting and acknowledging reminders.

Task 2: Implement a calendar view to navigate through scheduled doses.

Day 5: State Management and Combine Framework

Task 1: Manage app state for scheduled reminders and use data binding to reflect changes instantly when a medication is taken or skipped.

Task 2: Use the Combine framework to handle changes in medication schedules and trigger notifications.

Day 6: Lists, Animation, and Adaptivity

Task 1: Utilize list views in SwiftUI to show daily medication schedules and animate updates as medications are marked off.

Task 2: Adapt the app UI to different screen sizes and implement gesture controls for snoozing reminders.

Day 7: Advanced SwiftUI Topics and Core Data Integration

Task 1: Design an adaptive user interface that supports accessibility features like dynamic text sizing and voice-over.

Task 2: Integrate the app with Core Data to persistently store medication data and schedule information.

Submission Guidelines:

1. Ensure that each answer is clear, concise, and reflects an understanding of the core concepts.

2. Diagrams can be hand-drawn and scanned or created using any digital drawing tool.

3. Provide references for any external sources used.

Submit your work in a single PDF document by end of Module.

4. You must submit your code on gitlab by the end of next day.