**Course**: B590 / Spring 2025
**Submission Date**: May 07, 09:00 PM
**Team13**

**Members**:


- Rajesh Kumar Reddy Avula (rajavula@iu.edu)
- Sudeepthi Rebbalapalli (surebbal@iu.edu)


## About the app:

- StepWise is a step tracking app that helps users monitor their daily step count, distance walked (in km or miles) and calories burned. It continuously tracks steps, even in the background or if the user forgets to stop tracking.
- Built with SwiftUI, it features three main screens:
    - Today Screen
    - History Screen
    - Settings Screen
- Navigation implemented using MainTabView for seamless screen switching via tab bar.

## Interacting with the app:

### Today Screen

- Displays real-time:
    - Step count
    - Distance walked
    - Calories burned
- Includes Start/Stop buttons to control tracking manually.
- Step tracking begins when "Start" is pressed and ends with "Stop".
- Launch screen shown on app open before transitioning to Today screen.

### History Screen

- Allows users to review previous activity data.
- Features three toggle buttons:
    - **Daily** – cumulative stats for the current day.
    - **Weekly** – stats for the current week.
    - **Monthly** – stats for the month.

- Uses **Swift Charts** to display:
    - Hourly data for daily statistics
    - Daily data for weekly statistics
    - Week-wise data for monthly statistics
- Enhances clarity with visual bar charts.

### Settings Screen

- Fully implemented using **SwiftUI List** view with clear sections:
    - **Daily Reminders**: Enable/disable notifications.
    - **Step Goal**: Set daily walking goal.
        - Notification sent when goal is achieved.
    - **Units Selection**: Toggle between kilometers and miles for tracking distance.
    - **Reset Step History**: Deletes all historical step data from Core Data.

## Xcode and simulator details:

- Developed using Xcode 16.2.
- Uses Core Motion and CMPedometer for step tracking (requires iPhone/hardware).
- Used iPhone 13 with iOS 18.4.1 to test the application using hardware (CMPedometer) and Xcode simulator iPhone SE (3rd Generation) with iOS 18.3 for testing the application.

## Frameworks Used:

- Core Motion
- Core Data
- UserDefaults
- UserNotifications
- SwiftUI
- Swift Charts

## How did we satisfy the project requirements?:

#### MVC Pattern

- Implemented using folders for Models, Controllers, and Views.
- While SwiftUI does not enforce strict MVC like UIKit, we have organized files accordi ng to our understanding of the pattern.

#### Multiple Input & Output Types

- We make use of multiple forms of input and output elements throughout our app

- SettingsView.swift uses List view to render setting options.
- Screens implemented using separate SwiftUI views:
    - HomeView.swift (Today)
    - StepHistoryView.swift (History)
    - SettingsView.swift (Settings)

**Platform-Native Frameworks**

- Only native iOS frameworks are used.

**Persistent Storage**

- Implemented using:
    - Core Data – StepWiseData.xcdatamodeld
    - UserDefaults – StepDataController.swift
    - UserNotifications – SettingsController.swift

We utilized several frameworks covered in the B590 course to develop our app, including **Core Data** (StepWiseData.xcdatamodeld), **Core Motion** (StepDataController.swift), UserDefaults(StepDataController.swift), **UserNotifications** (SettingsController.swift), and **SwiftUI**. In addition, we integrated **Swift Charts** (StepHistoryView.swift) to visualize historical step data. **Swift Charts** was not introduced during the course, we incorporated it to enhance the user experience through clear and interactive data visualization.

## Changes from original app design

- Transitioned from UIKit to SwiftUI to enhance responsiveness and modernize the design.
- The previous version of the app had a less polished user interface and lacked responsiveness across various iPhone models and iPads. SwiftUI offered a more consistent and adaptive layout.
- SwiftUI's declarative syntax and powerful UI-building features enabled us to create a cleaner, more maintainable user interface.
- After working with UIKit in Homework 4, we opted to migrate to SwiftUI for better usability and maintainability.
- The transition impacted all screens of the app. While the business logic remained largely unchanged, we restructured the codebase to align with SwiftUI practices. Controllers, models, and views are now clearly organized into separate folders for maintainability.

## Noteworthy Elements:

**Background Tracking Logic:**

• Implemented using UserDefaults to ensure steps are tracked even if the user forgets to stop.

• At day-end, data is saved automatically to Core Data.

**Demo Data Generator:**

• The function createAndSaveFakeStepData() in StepHistoryController.swift generates and stores 30 days of sample data.

• Triggered using a UserDefault key when the user resets history in the settings screen