Golfer SRS

Prepared by: Jeffrey Martin

October 31, 2023

CSC 491 Project

1. INTRODUCTION

Purpose

The purpose of Golfer is to simplify and enhance the experience of golf enthusiasts by creating a mobile application that allows users to seamlessly manage golf scorecards, golfers, handicaps, and golf game setups. The primary goal of this application is to streamline and expedite the process of organizing golf games and determining extra strokes for players, reducing the time and effort required while minimizing the potential for errors.

Intended Audience

Golfer will be designed with a specific set of intended audiences in mind. Golf enthusiasts, amateur golfers. And golf course operators. Golfer will be designed to be user-friendly, catering to a diverse audience of golfers, from novices to seasoned players.

Intended Use

Golfer is intended to simplify the management of golf-related activities, specifically golf game setup, calculation of extra strokes per player, and persistent data management.

Scope

The scope of this application is keeping gameplay scores with variables like handicaps.

2. SYSTEM FEATURES AND REQUIREMENTS

TECHNICAL REQUIREMENTS

Software Tooling

This application will take advantage of Apple's open-source Swift programming language. It will be built with Xcode targeting the iOS operating system for iOS versions 17 and up.

IDE Requirements

Xcode version 15.0.1 and above will be required for debugging and running this application.

Operating System Requirements

This app will be designed for the iPhone using the iOS operating system. The minimum iOS version supported will be 17.0.

DATA REQUIREMENTS

Model

Game Document

Field	Type
players	Array of Player objects that represent any player that has been created within the app.
previousGames	Array of Game objects that represent any previous games. Used to calculate players handicaps
Courses	Array of Course objects that represent any course that came default in the app or was created by the user.
currentGame	An optional Game object that represents the current game being played or nil if nothing is active.

Player

Field	Type
id	UUID String - Unique id that represents the player
name	String - Name of the player
handicap	Integer - The handicap index for the player calculated by average of best 8 scores from 20 most recent games

Game

Field	Type
id	UUID String - Unique id that represents the game
startDate	Date - Date of when the game started
scoreCard	ScoreCard - An object that is responsible for keeping score of the game

Field	Type
courseId	String - The unique ID of the course this game is played on
endDate	Optional Date - Date for when the game is completed or nil if it isn't completed
currentHole	Int - The current index of the hole being played

Score Card

Field	Type
numberOfHoles	Int - The number of holes being played
playerScores	Dictionary - Keyed by player id with a value of a Score object
players	Array of Strings - A list of all the players on the score card

Score

Field	Type
playerId	String - Unique Id of the player who represents this score
holes	Dictionary - Keyed by hole index and the value is the score on the hole

Course

Field	Type
id	UUID String - Unique id that represents the course
name	String - Name for the course
slopeRating	Integer - The slope rating for this course

Field	Type
courseRating	Integer - The courseRating for this course
holes	Array of Hole Objects - Every hole that is on this course

Hole

Field	Type
id	UUID String - Unique id that represents the hole
Index	Integer - Index of this hole during gameplay
par	Integer - The number of shots it should take to get the ball in the hole

Persistent Data

- All data will be placed in a given GameDocument following a document data structure.
- This data will be saved to disk and loaded from disk using SwiftData. This is a new data technology from Apple built from CoreData.
- The operating system and SwiftData will manage the saving and querying of the data.

FUNCTIONAL REQUIREMENTS

- Game Creation
  - Create/Edit/Delete/Select Players
    - There will be a dedicated UI view that will list all players currently saved to disk.
    - This view will contain a search bar used for searching the player array.
    - This view will also contain a button used to add a new player.
    - The list will contain a row for each player showing the player's name.
    - Each row will incorporate Apple's swipe interaction to reveal a delete and edit button for each player.
    - The UI will allow up to 4 selections showing a checkmark on each selected player
    - The UI will also include a button to go to the next step.
  - Course Selection / Creation
    - There will be a dedicated UI view that will list all the courses currently saved to disk.
    - If no courses are on disk the application will load some default course data for quick playing.

    - This view will contain a search bar used for searching the available course array.
    - This view will also contain a button used to add a new custom course.
    - The list will contain a row for each course showing the course's name as well as the number of holes and the slope and course rating for the course.
    - Once the player taps on a row to select a course the UI will update and move into Game Play
- Game Play
  - General
    - The gameplay view will show all the data needed for the game. This will include a list of the players with scores, what hole is currently being played, all of the hole data, as well as the ability to advance holes and update the score.
  - Hole Selection
    - The main UI will have two buttons. One button will be to advance to the next hole or complete the game if we have advanced through every hole. The other button will be to go back to previous holes if needed until you are back on the first hole.
  - Score Entry
    - The main UI will have a list containing every player currently playing in the game.
    - On tap of the user, the UI will open a field to enter the number of strokes it took for the player to get the ball in the hole.