ScoreSnap App

App Description

Summary

This app is designed to help youth players, their parents, and coaches track important sports metrics over time. The initial version will focus on youth basketball. For this version, the app will track end-of-game score and the win/loss record for a player's teams. The app tracks basketball game scores based on photos of the scoreboard, which are uploaded to AI for image processing and OCR. The app will list out game results, present the win/loss records for the season. The user can track multiple teams. People will want to use this app because it is quick and easy.

Detail

Users will use this iPhone app to track a basketball player's record based on photos of end-of-game scoreboards. The user chooses to either take a photo from the in-app camera, select a photo from the library, or manually enter end-of-game score results.

The app uses AI-powered image processing to identify the scoreboard in the photo and optical character recognition to determine the score of the game. The app then prompts the user to verify the final score, gives the user to ability to the score, has the user to select whether the player's team had a Win or a Loss, and enter the opponent team name. Game Date, Game Time, and Game Location are pulled from the photo's metadata. If location is not available in the metadata, leave this field blank. GameAddDate and GameAddTime are the date and time the Game Info was added to the app.

Note that this first version will only support 1 sport (basketball). However, the app should be designed to support additional other sports in future iterations. Furthermore, this first version tracks a minimal amount of game info. Future iterations make track more detailed information.

Goals

Business Goals

- Simplifying the process of tracking scores at the end of basketball game, as the scoreboard often gets reset quite

quickly.
· Provide an easy way for players, parents, coaches, and fans to track the season win-loss-tie records for their players.
· Ensure the service provides reliable tracking of data but is also profitable enough to enable scaling and additional feature development if the product is popular.
· Create a product that has enough convenience and value that user would pay a recurring subscription.
· Market this product to the hobbyist segment of the youth basketball community that wants simple and easy-to-use app that enables tracking of the top-level metrics for the season (wins vs losses for the season, end-of-game scores, location of game).

User Goals

· Have an easy-to-app that tracks a player's win/loss/tie record during a season. Track the individual game results.
· Be able to see record and results for a player's multiple teams.
· Be able to toggle between multiple players.

Development Goals

The app will be developed in 3 stages.

· V1.0 version is the proof of concept. It contains all critical elements to determine whether the design is function and whether the app has value. There will be 1 user. All information stored in core data and there will be no subscription features.

· v2.0 version is the beta testing version. The app will be productized for distribution to a dozen users for testing and feedback. The app needs to have proper security. All information stored in core data and there will be no subscription features.

· v3.0 public launch of the app. The app is launched in the Apple App Store. The app needs to be ported over to supabase for account management, data storage, and usage statistics. Additionally, the app will use revenueCat to manage subscriptions and track user conversion.

Assume all features in this document are for v1.0 unless labeled as #v2 or #v3.

Additionally, there will be a v4.0 – this would be expanding the app to significantly more detailed basketball metrics. It could also include expanding the app to additional sports. These requirements and features will be tagged #v4.
Data Architecture
Data will be managed with iOS Core Data.

A proposed Data Structure is:

- Player
    - Player Color this is strictly for visual differentiation
    - Display Order
    - Sport
        - Team
            - Team Color (this is strictly for visual differentiation)
            - DisplayOrder
            - Team Record (Wins - Losses – Ties, calculated from game info and updated when there is a new upload for the team)
        - Game Info
            - GameDate
            - GameTime
            - GameLocation
            - Win/Loss/Tie
            - TeamScore
            - OpponentScore
            - OpponentName
            - Notes
            - GameEditDate
            - GameEditTime

Note that this first version will only support 1 sport (basketball). In this version, Sport is set to “Basketball”. The user should never see or be able to change the Sport setting.#v4 - However, the app should be designed to support additional other sports in future iterations. Other sports may have different additional fields under Team Record.   Future iterations will allow selection.

This first version tracks a minimal amount of game info.

- #v4 - Future iterations make track more detailed information, such as the player's minutes played, points scored, three-pointer attempts, three-pointers made, etc.

This version of the app will support multiple players. Each player can have multiple teams. Each team will have its own Team Record.

By default, the app is setup to display information for one player/child playing on one team. However, the app allows the user to add more than 1 team. The multiple team option is necessary for players that play on multiple teams.

Add UUIDs for all Core Data entities to ensure unique identification and support future cloud sync. Select a structure to enable easy migration to supabase in #v3.

Features

Setup Workflow

The app will have a setup workflow that walk the user through setting up the first player and team, as well as grant permission to access the camera and photo library.

Player Management

User can add players, remove players, reorder players. User can add teams associated with players. User can add/remove/reorder teams. Reordering affects display order in other views. User can change the default color associated with a team using swiftUI colorpicker.

Upload GameInfo – camera

Use camera to capture information for AI processing and add a gameinfo record.

- User launches an in-app camera to take a photo of the game scoreboard.
- Once the photo is taken, the user see preview must accept or

cancel. If they cancel, the user it taken back to the camera.

- If the user accepts, then the app calls an AI service to determine the score from the photo and return the score.

- “Use AI response and user input to create Game Info” (see feature details below)

Upload GameInfo – photo library
Select photo from photo library for AI processing and add a gameinfo record.

- User pull ups the photo library
- User selects a photo from the library.
- Confirm that the upload is a photo and not a video.
- “Use AI response and user input to create Game Info” (see feature details below)

Use AI response and user input to add a GameInfo record
Call an AI service for image processing and optical character recognition to extract a game score from a photo. Then solicit user input on additional details to complete a game info record.

- Determine the final score using AI optical character recognition.
  - o Step 1: User selects outcome first [Win] [Loss] [Tie] buttons
  - o Step 2: Show AI extracted scores neutrally "Game Score: 65 - 72" There is no assumption about which is the teamscore vs opponentscore.
  - o Step 3: Allow the user to click to edit the score, if needed.
  - o Step 4: Based on selection, clearly label the scores
    - § If Win selected: "Your Team: 72 Opponent: 65" then higher score automatically becomes "Your Team"
    - § If Loss selected: "Your Team: 65 Opponent: 72" then lower score automatically becomes "Your Team"
    - § If Tie selected, confirm that both score are identical.
- Allow user to enter the opponent name.
- User must select the player and team for which the game info will be associated. Default to current player and current team.
  - o The user has the option to select a different player and different team for which gameinfo is being uploaded. This selection affects the global context of “current player” and

"current team".

- Extract the game date, game time, and game location from the photo metadata.
    - If any of these are missing, use the date, time, and location at the time of upload. If location is still not available, leave this field blank.
- Store the teamscore, opponentscore, win/loss/tie, game date, game time, GameEditDate, GameEditTime, and location values.
- A future iteration will store photos in the cloud (supabase) for training AI. In version 1.0, stub out this code.
- Return user to the screen when "add upload" was clicked.
- Use good API-key security practices
- Build API call limitations per user to ensure the calls to Claude are not being abused. Limitation should be coded in a way that is easy for the developer adjust. The API restrictions should not be visible or editable by the app user.

Upload GameInfo – manual

Allow a user to add a game info record.

- User is presented a screen that allows them to:
    - Selects outcome first [Win] [Loss] [Tie] buttons
    - Enter Score
        - Team: XX
        - Opponent: YY
    - User must select the player and team for which the game info will be associated. Default to current player and current team.
        - The user has the option to select a different player and different team for which gameinfo is being uploaded.
    - Enter opponent name
    - Enter game date, game time, and game location
        - Default to the current date, time, and location

Edit Game Record

Allow user to change a gameinfo record.

- Allow user to edit all Game Info fields, except GameEditDate and GameEditTime.
- GameEditDate and GameEditTime are not visible or user-editable fields.

Views

Home View

Home View displays the Team Records and the summary Game Info for a single player and single team.

- On setup, the Home View will default to the one player and one team.
- If there is more than one player or team setup, the Home View will default to the last viewed player and last viewed team.

User can change players using segmented controls at the top. The app will have segmented control across the top so all players visible at once. Ordering of players is based on ordering in the Players View.

- V3.0

User can change the player. User can change teams associated with the selected player. Team selection will be a dropdown

Players View

The Players View allows the user to, add/remove players, reorder players. The Players view allows the user to add/remove/reorder teams. The Players View allows the user to select or change the color associated with a team.

The View has a Players section and a Teams section.

In the players section, the current player is highlighted. All teams for the current player are displayed in the Teams section.

User can select a different player to highlight. The player becomes the current player. The list of teams for the current player is updated in the Teams section. The last viewed team for the player becomes the current team. If there is not last viewed team, then default to the first team for the player.

User can added a player by pressing a plus icon next to Players heading
User can re-order players.

User can tap an info icon next to the player name to edit player details and to delete a player.

User can add a team by pressing a plus icon next to Teams heading
User can reorder teams.
User can long press a team to edit team details.
User can tap an info icon next to the team name to edit player details and to delete a team.

Games View
The goal of the Games screen is to show information for all teams for current player. Teams are listed based on display order. User can scroll down through all teams, seeing each team, team record, and games. The Games View has more detailed game information than the Home View. Games view is also where the user can tap to see the information icon. Clicking the information icon allows the user to edit Game Info.

Game EditView
GameEditView lists all games for a given player, organized by team.
Detailed game info is provided.
User can tap on an info icon next to a game to allow the user to edit Game Info.
Team are listed based on the team display order.

Settings View #v3
This view and functionality will be stubbed out in the v1.0 version (proof of concept) and v2.0 version (MVP and beta testing) of the app.

- #v3 - Users can view their account information (username, password, unique ID) and manage their subscription.
- #v3 – User can set the default LengthofTimer

Navigation and UI
Global State
The app should have a global understanding of the “current player” and “current team” across views.

- **Home View:** user selection changes the current player and current team

- **Upload GameInfo – camera:** user selection during the upload process changes the current player and current team
- **Upload GameInfo – photo library:** user selection during the upload process changes the current player and current team
- **Upload GameInfo – manual:** user selection during the upload process changes the current player and current team
- **Example 1:**
  - User is viewing Player1, Team1A in the app in Home View. However, they are at a game for Player2 and launch the in-app camera to take picture of the scoreboard. They are in the "Upload GameInfo – camera" workflow.
  - During the "**Use AI response and user input to add a GameInfo record"** workflow, the user selects Player2 and Team2A for the upload information.
  - Upon completion of "**Use AI response and user input to add a GameInfo record,"** the user is returned to the Home View. Home View now present information for Player2 and Team2A. The Home Views display include the recently added record for Player2.
- **Example 2:**
  - User is viewing the Home View with information for Player 1, Team1A. The User switches to Player 1, Team1B.
  - The User goes to "**Upload GameInfo – photo library**" to add a scoreboard photo. The "**Use AI response and user input to add a GameInfo record"** screen defaults to Player 1 and Team 1B for the gameinfo record.

Bottom Navigation Tabs

- Home View
- Games View
- Players View

Floating Action Menu

A large floating action button appears in the bottom right corner of Home, Games, and Players views.

- Tap button: instant camera with current player/team context to take photo and go into "Upload GameInfo – camera"
- Long-press the button to show options menu (camera/library/manual)

Top Navigation

- Gear Icon – floating button it the upper right to access Settings View

Functional Requirements

Priority 1 (v1.0 features)

- Use Core Data to store information
- Include all the correct p.list files to enable access to camera
- Setup workflow
  - Setup the first player
  - Setup the first team
  - Help user update settings to allow access to the camera and photo library
- Home View
  - The goal of the home screen is to show information for the current player's current team.
  - Allow user to switch players and switch team for the current player.
  - Display team record for the selected team
  - Display gameinfo for the selected team
- Upload workflow (3 versions)
  - User can click button to take a photo using the in-app camera, call the AI service for image processing, allow user input to complete the gameinfo records, and save the record.
  - User can click button to select a photo from the photo library, call the AI service for image processing, allow user input to complete the gameinfo records, and save the record.
  - User can manually enter a gameinfo record
- API security
  - Use good API-key security practices
  - Build API call limitations per user to ensure the calls to Claude are not being abused. Limitation should be coded in a way that is easy for the developer adjust. The API restrictions should not be visible or editable by the app user.
- Players View
  - Allows user to add players, remove players, and reorder players.
  - Allows user to add teams, remove teams, and reorder teams

for a player.
- o Allows user to change color-coding for teams.

· Games View
- o The goal of the Games screen is to show information for all teams for current player.
- o Uses same player selection UI as Home screen (segmented control/More button)
- o Player selection changes global context and resets to first team in the display order for the current player.
- o Show records for all teams, organized by team display order.
- o For each team, show:
  - § Team Name + Team Record
  - § A table of select Game Info
    - · GameDate
    - · Win/Loss/tie
    - · Score (bold the playerscore number, no bolding for ties)
    - · OpponentName
    - · Location (only display City)
    - · Notes
- o When user taps on a row in game info, display an information icon.
  - § If user taps on the information, take them to a GameInfoView that displays and allow user to edit all Game Info fields, except GameAddDate and GameAddTime. GameAddDate and GameAddTime are not visible or editable fields.
- o If user swipes left on a row in game info, provide the trashcan icon to allow the user to delete the GameInfo from the database. Update team record appropriately.
- o Have a floating "camera" icon to take user to in-app camera to begin Add Upload workflow.
- o Allow a user to manually add a game for the current player.

Priority 2 (for #v2, a minimum viable product ready for beta testing)
Add code for priority 2 features but have them stubbed out. Label the Priority 2.

- · Revenue model

    - Future version will incorporate services from revenueCat to handle subscriptions and paywall
    - User must enter credit card info to get 2-week free trial
    - Automatic subscription enrollment after 2 weeks
- Ability to assign a game to a different player and team

Priority 3 (for #v3, which are refinements to be made prior to launching to the app store)

- Settings View
- Migration to Supabase for account management, data storage and backup, and usage metrics
    - Store uploaded photos for training of AI
- App store feedback reminder

State Clarification Examples

Mid-upload context change

- User starts on Home View: Player1/Team1A
- Takes photo via floating action button
- During upload, changes to Player2/Team2A
    - Upon return to Home View, information should reflect Player 2/Team2

Player View

- Adding players and teams in players view does not change global context

Player deletion

- Current contact is Player 2. User deletes Player 2. Global context defaults to the first remaining player.
- If there are not remaining players, display “add player”.

Team deletion

- Current contact is Player 2/Team2B. User deletes Team 2B. Global context defaults to the first remaining team for Player 2.
- If player 2 has no remaining teams, display “add team”

Architecture Details

- Use SwiftUI
- Use the API from Claude AI to determine which values in the photos represent the final score in the game.
- When running the determination note that:
    - § Basketball scores are usually whole numbers that range from 0 to 200.
    - § Basketball scores do not have the ":" character.
    - § End of game scores will show a clock at 00:00, or near 00:00.
- Data will be stored locally in this first version of the app, with the exception of API calls to Claude AI for image processing and OCR.
- No plans to support Android. This is an iOS app.
- Follow Apple's Human Interface Guidelines for gestures, controls, app icons, typography, and design patterns: https://developer.apple.com/design/human-interface-guidelines
- Plan to include revenuecat for future subscription management.
- Plan to include supabase for future cloud database infrastructure.

Warnings

- We're focusing on functional accomplishments of features in this stage, not designing UX in extreme detail
- If a feature or tech choice seems ambiguous, ask me for clarification such that you would get what you need to continue
- You should consider how tech choices may evolve or change if the application scales and give me recommendations with tradeoff consideration
- We should have a clear architecture for the app, including main infrastructure considerations, services/microservices required, critical 3rd party APIs choices, etc

Technical Specifications

- Minimum iOS Version: iOS 16.0+ (for latest SwiftUI features)
- Orientation: Portrait only
- Network Requirements: Internet connection required for AI

calls. Will need internet connection for user authentication and database calls in post-

- Permissions Required:
  - Photo Library access (read only)
  - Network access for Claude AI API
- Upon installation provide a workflow that guides the user to allow photo library access
- Add UUID primary keys to all entities

Error States

- Network failure: display message “Cannot connect to network for image analysis” and then offer manual entry
- Invalid JSON: Retry once. If there is still an invalid JSON that display error message “JSON error” and then offer manual entry
- Rate limit: display error message “Rate limit error” and then offer manual entry
- No access to Photo Library: display error message “No access to photos” and then guide user to enable access.