FINAL PROJECT REPORT

Francisco García Martínez
Hrishikesh Varma

Name of the project application
Coach Canvas

Primary Supervisor
Evin McCarthy

Name and student numbers of group members
Francisco García Martínez (D24126042)
Hrishikesh Varma (D24125776)

Link to Trello board
https://trello.com/b/mrbs1FiR/coach-canvas

ACKNOWLEDGMENTS

We sincerely thank each of our fantastic lecturers over the duration of the course: Keith Gardiner, Brian Vaughan, Niamh O'Hora, and especially, our supervisor Evin McCarthy, whose guidance during the development of this Final Project kept us on track and has been of immense help.

Every one of our coursemates, whom we have continuously been learning from inside the classroom, and have become good friends outside of it.

Our loved ones, families and friends, from our home countries and from Dublin, whose support has been crucial in pushing us to this point.

And finally, our testers from local clubs and everyone who volunteered out of good will and passion for our project. Their contributions and feedback are immensely appreciated, and will continue to shape Coach Canvas.

DECLARATION

We, Francisco García Martínez and Hrishikesh Varma, hereby certify that the material which is submitted in this final report, towards the award of a Masters of Science in Creative Digital Media & UX, is entirely our own work and has not been submitted for any academic assessment other than part-fulfilment of the award named above.

Date:
30th November 2025

Signed,

Francisco García Martínez
D24126042

Hrishikesh Varma
D24125776

CONTENTS

LIST OF FIGURES

GLOSSARY OF TERMS

Manager Mode: The role in the app with elevated permissions to create clubs, approve join requests, create events and edit lineups.

Player Mode: The role for team members who can join clubs via code, set availability and view lineups.

Club: In football, it's the entire organization behind a team, which includes the players, staff, facilities, and supporters. Within the Coach Canvas database, it's a logical grouping composed of a coach and players.

Join code: A short alphanumeric token or generated by a club for players to join easily.

Event: A scheduled item in the app representing a Match, Training or other team activity.

Availability: The status a player sets for an event (available / unavailable / doubtful), visible to managers for selection decisions.

Lineup: A saved arrangement mapping pitch positions to players for a specific event or template.

Formation: Tactical arrangement (for example 4-4-2, 4-2-3-1) used to position players on the pitch canvas.

Bench / Substitute: Players not selected in the starting lineup but available for substitution.

Game Mode: The match-level UI that allows managers to record real-time match events such as goals, assists and cards.

Tap-to-swap: The default interaction implemented to exchange two players' positions by tapping, chosen for cross-device reliability.

Drag-and-drop: An optional interaction model for repositioning player cards by dragging; proposed for devices with better input affordance.

Firebase: Google Cloud Firestore, the NoSQL document database used for storing users, clubs, events, lineups and related data.

React Native: The cross-platform mobile framework used to build the Android app.

Cloud Messaging (FCM): Firebase Cloud Messaging, used for push notifications to devices.

Service layer / service modules: Encapsulated backend access objects (for example clubService, eventService) that implement domain logic and Firestore access.

TypeScript: The strongly typed superset of JavaScript used across the codebase to reduce runtime errors.

SOA (Service-Oriented Architecture): Architectural style in which application functionality is organised into independent services.

Skeleton loader: Lightweight placeholder UI shown while data is being fetched to improve perceived performance.

MVP (Minimum Viable Product): The core deliverable scope with essential features required to validate the product idea.

Microcopy: Small in-context text used to clarify actions, reduce errors and guide users.

Onboarding: The initial flow that helps users sign up, select a role and join or create a club.

A/B test: A controlled experiment to compare two variants of an interaction to determine which performs better.

Jest: JavaScript testing framework used to run automated unit tests on portions of the code.

APK: Android application package produced for testing on devices.

SVG / Iconify: Scalable vector graphics and an icon set used for lightweight, resolution-independent UI icons.

Trello: The project management board used to track tasks, sprints and epics.

1. INTRODUCTION

This document has been written to detail the creation process of Coach Canvas, a lightweight Android-based mobile application designed to support grassroots football managers and players in planning tactics, managing squads, and scheduling matches and training sessions. The purpose of this report is to provide a comprehensive record of the project, from initial concept to finished artifact, through research, design, prototyping, testing and evaluation.

Coach Canvas was conceived to address persistent inefficiencies experienced by grassroots teams. At local and amateur levels, managers and players typically rely on a heterogeneous mix of whiteboards, paper lists, group messaging apps and spreadsheets to communicate tactics, confirm essentials like availability, and coordinate fixtures. These tools are functional but fragmented, and they impose coordination costs that distract from the fun parts of the game: coaching and playing. Coach Canvas occupies a middle ground between heavyweight analytic suites used by professional tactical teams, and old-school manual methods. The application aims to be simple enough for a weekend team to adopt easily, while offering enough structure to meet the needs of semi-professional and serious amateur clubs.

This report will document how our initial aims were translated into design objectives, and how the designs were validated and iterated through user-centered methods.
The document is organised into five main sections. Having introduced the purpose and scope of the project, we will follow up with User Needs Analysis, defining the project's aims and objectives more fully, and present the primary user personas that emerged from our research.
The third section, Background Research, situates the project within relevant literature and competing products within the current market, reviewing related applications, UX patterns in sports software, and technical considerations for the field.
In Methodology & Approach, we explain the chosen development and design methods, including types of prototyping, art direction and visual language, technologies and approaches used throughout the development and how they affect the different sections of the application.
The fifth section, Testing & Analysis,presents the results of the research work, including all sessions performed, findings, and most importantly, the design changes motivated by user feedback.
With the Evaluation section, we aim to assess the finished prototype against the project objectives and our own perceptions, reflecting on technical achievements and user acceptance.

2. USER NEEDS ANALYSIS

2.1. Introduction

This chapter defines and analyses the user needs that shaped the design and development of Coach Canvas. It explains the project aims and objectives, documents how those objectives were grounded in direct field research and iterative prototyping, and presents the personas and scenarios used to validate design choices. The structure of the chapter is as follows:
First, it clarifies the approach to user needs gathering and summarises the qualitative data gathered through interviews, observational sessions and five prototype test iterations. Third, it presents the core personas that guided design decisions and maps their requirements to concrete functional features. Fourth, it restates the project aims and links them to prioritized user needs. The chapter closes with a concise conclusion that synthesises the principal design implications and next steps.

2.2. User Needs Research

Our user research strategy prioritised in-context, qualitative investigation with real users. This decision followed the project objectives to produce a practical, usable tool for grassroots teams and to iterate quickly using low-cost prototypes. The research sequence consisted of semi-structured interviews with coaches and players, observational sessions at training, five rounds of prototype testing of increasing fidelity, and short ad-hoc field tests with local clubs. The prototype iterations were: sketches/paper prototype, low-fidelity interactive wireframe, mid-fidelity coded MVP focused on lineups, a functional coded app with login and basic scheduling, and a fully fleshed out Figma prototype. Each round elicited targeted feedback about discoverability, interaction, feature priorities and real-world constraints such as device type and time pressure.

The evidence base for user needs therefore consists of qualitative findings from those sessions. Representative data points include feedback sessions recorded during prototype testing, and these qualitative data points have been used to prioritise scope and refine our interaction models and user needs. The key insights that we found were the following:

- ***Single hub for multiple tasks***
  Coaches find current workflows fragmented, including WhatsApp groups, printed lists, whiteboards and spreadsheets, which are commonly used in combination. This creates duplication and communication errors. Our users strongly value a single, lightweight hub that consolidates schedule functionalities, squad management and quick tactical sharing.

- ***Simplicity over depth***
  Coaches want tools that are quick and reliable. Many users explicitly rejected complex apps (which we analysed as competitors) for their usability difficulty. The

tactical aspect should allow fast lineup creation and sharing that can be completed in minutes. Detailed blackboard-style animation or deep analytics is desirable to a minority, but must not get in the way of the primary, rapid workflows.

- ***Availability is critical***
  Player availability is repeatedly cited as the single highest friction point. Managers need an accurate, one-glance summary of who is available, updated in near real time. Several managers asked for automatic attendance summaries to avoid late phone calls the morning before a game.

- **Clarity of communication**
  Players placed high value on clear, visual representations of their roles, especially at lower levels of football. Staying "in the loop" about what the manager wants players to do on the field was reportedly very important, with accessible visuals that clearly show where they will be playing and what their role in the lineup will be. However, a simple pitch view was judged more useful than long textual briefings. Players also want lightweight personal statistics and a record of performance.

2.3 Personas & Scenarios

This level of insight expanded our user model ideas to paint a clearer picture of possible user personas, which condense the research into practical archetypes to inform design decisions. The following personas were used throughout the project to validate features, workflows and scenarios that Coach Canvas can suit.

1. **Alex | Amateur coach**

- Aged 32, part-time coach for a semi-professional team.
- Manages training sessions, develops tactics, and communicates with players via WhatsApp and notebooks.
- Values simplicity, quick access to information, and reliable scheduling.

Use cases:

- Alex opens the app on Monday morning, and schedules training drills for Wednesday. On Tuesday, he checks player availability to suit the training session to the players participating.
- On Thursday, he uses the Schedule function to schedule the next match and looks at the Squad to check on his players, thinking about the game.
- For Sunday's match, he uses the Upcoming Match section to adjust his formation and lineup to the players who are reported to be available. When he's finished, he marks his tactics as ready to view.
- During the game, he records the match events and statistics, which are shown afterwards on the players' profiles in the Squad section.

2. **Priya | Part-time player**

- Aged 24, she works full time in a Junior role, trains once a week and plays for a lower local division football team on the weekends.
- Needs clear, accessible information on match lineups, training drills, and availability deadlines.
- Seeks an intuitive, quick-usage interface that fits into her limited free time.

Use cases:

- On Friday, Priya gets a notification for the upcoming match, checks the Schedule tab, and updates her availability status.
- On Saturday, she views the coach's lineup in the Next Match tab. She can see her position and any tactical notes before arriving at the pitch on Sunday morning.
- After scoring a goal in Sunday's match, she gets a notification on Monday that her statistics have been updated. She looks up her own profile and checks on some of her teammates' progress in the Squad tab.

Establishing these user personas has helped to shape our conceptualisation of Coach Canvas, influencing treatments and prototypes. In result, some features such as the split user model of the application, allowing for a Player Mode with read-only tactical views and availability toggle. To balance app uniformity and prevent confusion, we settled on a unified Home screen, prioritising quick access to Schedule, Squad, and Next Match functions.

2.4 Aims & Objectives

The main aim is to provide a lightweight, single-hub mobile application for non-professional football managers and players to plan tactics, manage squads and schedule events. The app should:

- Support a compact set of core workflows (create and share lineups, manage squad availability, and schedule events).
- Offer an intuitive lineup editor that enables quick tactical creation on mobile via simple taps.
- Implement a simple club join model, and allow clubs to persist independently of individual managers.
- Provide tactical presets and easy generation of new tactical setups.
- Prioritise low-friction onboarding and minimal required account setup for players.
- Preserve player privacy and support low-connectivity core flows.

2.5 Conclusion

The user needs analysis confirmed the original aims, that grassroots coaches and players require a compact, reliable tool to reduce the noise of “analog” communication and to make match preparation faster and clearer. The research demonstrated a clear hierarchy of needs (availability tracking, quick lineup creation and schedule visibility). Iterative prototype testing validated that small, well-scoped features deliver disproportionate value in the target context.

This chapter has documented how empirical observation and prototyping transformed broad aims into concrete design objectives and a prioritised feature roadmap. The needs-driven approach kept the project grounded in the realities of grassroots football and ensured that the core product decisions are defensible, testable and aligned with user value.

3. BACKGROUND RESEARCH

3.1 Introduction

This chapter situates Coach Canvas within the current landscape of coaching and team management tools, describes the external inspirations that shaped the design approach, and distils the strategic conclusions drawn from competitor analysis. It then reviews the technological choices that underlie the prototype and defines the concrete design requirements that followed from the research. The goal is to demonstrate why the selected feature set, interaction patterns and technical stack are the appropriate responses to the needs of grassroots and semi-professional teams, and how Coach Canvas seeks to position itself against existing products.

Aspiration and Non-Domain Influences

Coach Canvas draws inspiration from a mix of domain and non-domain sources. From productivity and design tools such as Trello and Figma we borrow direct-manipulation and card metaphors, which simplify complex arrangements into intuitive drag, drop and tap interactions. From health and fitness apps we adopt the single-screen dashboard approach that makes the most important actions immediately visible. Educational apps such as Duolingo provide lessons about progressive onboarding and contextual micro-tutorials, useful to lower the barrier for users unfamiliar with tactical software.

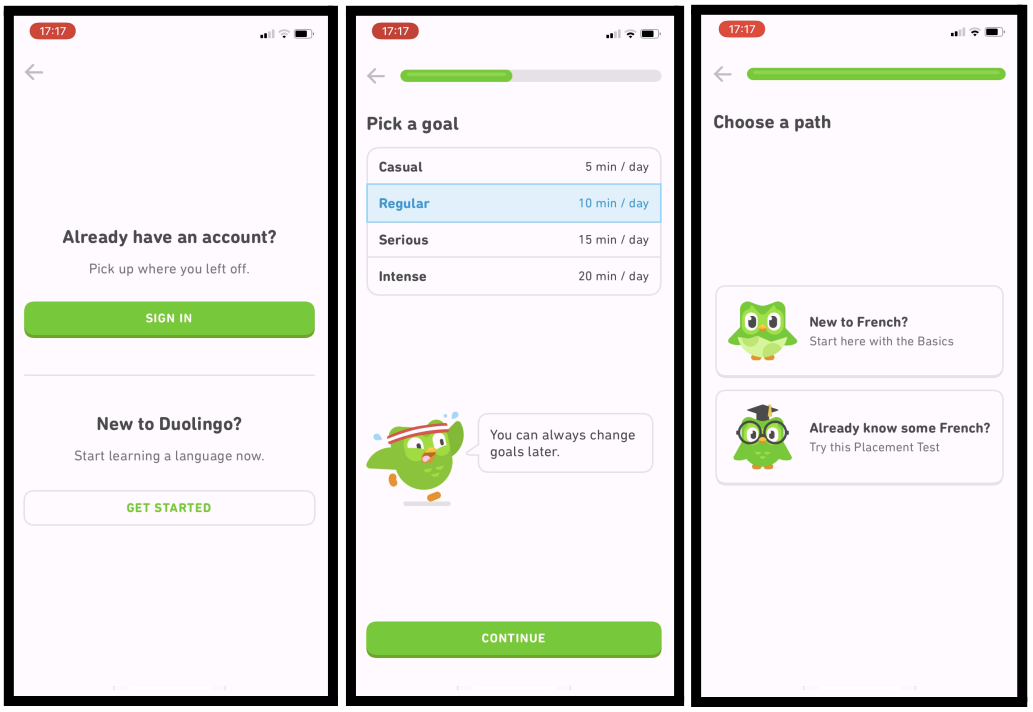

Figure 1: Duolingo onboarding flow

Mainstream football games suggest compact visual conventions for player cards and squad overviews. These influences are not decorative, but they shape concrete

interaction choices (a prominent home hub, card-style player representations, reliable and accessible behaviors, and accessible data visualisation).

Figure 2: EA Sports FIFA 24 player card design

Competitor analysis

A structured review of direct competitors revealed a clear set of strengths and weaknesses in existing products, and helped locate a gap in the market that Coach Canvas can address.

Coach Tactic Board

1. ***Analysis***

    a. ***App functions***
    Coach Tactic Board is a mobile application aimed at football coaches for, primarily, designing tactics and planning training sessions. It is positioned as an essential tool for pre-match preparation and drill design, offering a visual and interactive alternative to traditional whiteboards. The app's functionality is rooted in providing tactical clarity, offering users a customisable digital canvas for drawing movements, adjusting formations, and building drills using virtual equipment such as cones, ladders, and mannequins. Teams and players can be customised with attributes such as names, numbers, and photos, which enhances personalisation and repeat usability across sessions.

    b. ***Flows***
    From a user flow perspective, the application operates in a relatively linear manner. Coaches can navigate from the home screen into specific modules for tactics or training, with access to equipment and players through a top toolbar. Player placement and movement are done via touch

interaction, with annotations created using various line types (solid, dashed or arrows). Team and tactic data are saved locally, and while tactics can be exported as PDFs or images, the process lacks online data synchronization or collaboration, which limits interactivity.

c. ***UI and style***
In terms of its design and visual language, Coach Tactic Board prioritises functionality over aesthetic innovation. The interface is very densely packed with features, which can be overwhelming for new users, especially those unfamiliar with icon-heavy UIs. This utility-focused approach may appeal to experienced coaches but could create usability barriers for casual users or those with lower digital proficiency. While users may appreciate the depth of tactical creation, they might beg for a more intuitive experience and smoother workflows especially when dealing with frame-based animations for drills, which can be quite time consuming and difficult to edit.

d. ***Type of use***
Coach Tactic Board is most primarily a single-session app, and mostly involved in its developer intent, as it takes a relatively long usage time to map out and plan tactics, training sessions and animations, given the very manual and highly customisable approach of the app.

e. ***Data and content requirements***
In terms of data and content, the application relies heavily on manual input. Users must create all tactics, lineups, and player details from scratch, although templates are provided for common formations. There is no integration with external databases or automated content generation, which requires a high level of initiative and ongoing customisation from the user.

2. ***User Reviews & Feedback***
A significant number of users have complained about technical issues such as unexpected crashes, bugs within the UI and glitches. The app having a paid mode also raises some concerns amongst free users, who claim that the app falls short in terms of tactical settings that you can save. Some other users, who might not be as tech-savvy, report struggling and finding the process of making a squad tedious and laborious.

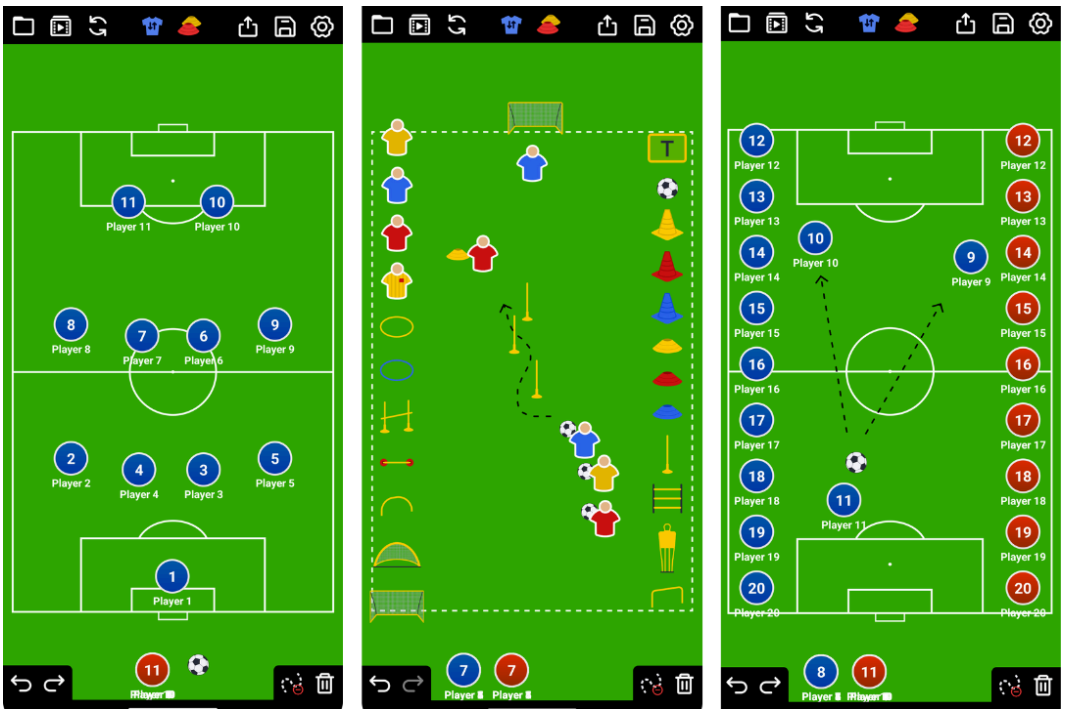

Figure 3: Coach Tactic Board UI

Coach Amigo

1. Analysis

a. App functions

Coach Amigo positions itself as an all-in-one management app for grassroots football, combining matchday planning, club statistics, and squad administration within a single interface. Its core functions center on a dashboard that aggregates upcoming fixtures (both matches and training), allows customisation of teams, and presents club-level statistics. It also supports management of multiple clubs from one account and provides basic roster controls, adding or deleting players and assigning roles.

b. User flows

From a user-flow standpoint, Coach Amigo's architecture appears unintuitive. Test users report difficulty in navigating between the various modules, finding the sequence of actions to, for example, schedule a new training session or edit club statistics, to be overly complex. Without a clear, linear progression through tasks, first-time users often become disoriented, unable to locate critical functions without repeated trial and error.

c. UI and style

Visually, the app suffers from a clustered layout and a lack of visual hierarchy. Multiple overlapping menus and conflicting options create cognitive load rather than reducing it. Elements that should be prominent, such as "Create Match" or "View Squad", are buried among secondary

controls. The overall aesthetic is utilitarian but neglects established UX principles around spacing, affordances, and clear signposting, resulting in what many describe as "bad UX."

d. Type of use

Because of its more live-focused nature, we could classify this app as partly single use, and partly multi-session. Setting up the team and pre-match preparation might require significant times of usage, whereas recording events during a match is designed to be a quick activity for which you might only need to open the app for a few seconds at a time.

e. Data and content requirement

The app requires extensive manual input: coaches must log every event in real time or upload match data afterward. While the base app is free, many advanced features, such as the attendance requests center, desktop app sync, and enhanced sharing options, are gated behind a Premium subscription (€3.75 per month). There is no integration with external calendars or databases, and the forced team-setup at first launch often frustrates users who wish to explore the app organically.

User data is stored in the cloud for synchronisation across devices, but there is no specialised tactical content library or drill database.

2. *User Reviews & Feedback*

Unfortunately, the app is not popular enough to have a significant chunk of reviews to analyse. There is one instance of somebody recommending it to an aspiring football coach on Reddit.

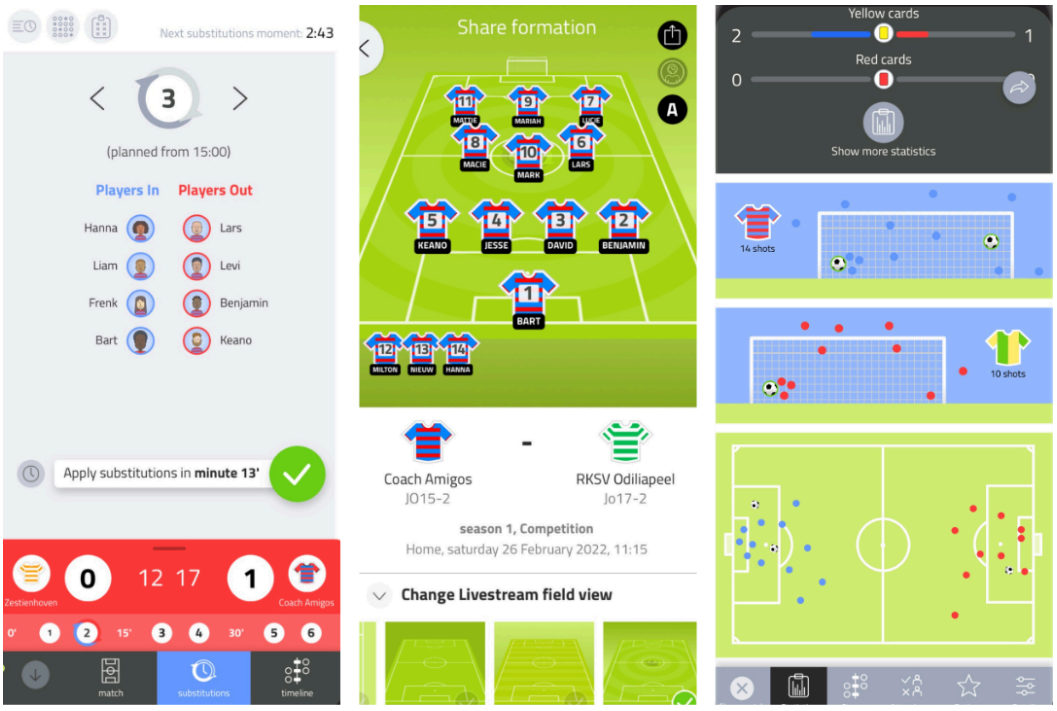

Figure 4: Coach Amigo UI

bCoach

1. ***Analysis***

    a. ***App functions***

    The bCoach app is designed specifically for football coaches, aiming to serve as a comprehensive tool for managing training sessions and planning tactics. It enables coaches to draw tactical formations and organise training content tailored to their team's structure. A standout feature of the app is its in-game mechanics, which allow real-time inputs during matches such as recording goals for both teams and noting events like offsides, fouls, red cards, yellow cards, goals, assists, and corners for individual players.

    b. ***User flows***

    The very first use of the app begins with an introductory information screen that clearly states the application is intended for coaches only. Following this, users are prompted to sign up without the option of using Google authentication. During the signup process, users must also select the number of players in their team, which is a mandatory step before proceeding. Once logged in, users can access the app's core functionalities. The navigation throughout the app is somewhat inconsistent, with a primary horizontal layout that occasionally shifts to a vertical view during pop-up interactions, leading to some confusion during use.

    c. ***UI and style***

    From a design and usability standpoint, the app faces significant issues. The interface is not user-friendly, and many elements are too small, making the content hard to read and interact with. This poor UI design makes it challenging to make modifications or work efficiently within the app, especially in fast-paced scenarios such as live match situations. Overall, the app's design limits its potential despite its useful features.

    d. ***Type of use***

    It's primarily a multi-session app, which coaches might have to keep using in short periods during matches or training sessions, whether to track statistics or to refer to tactics when instructing their players. However, there is a component of single-session usage when setting up the entire team in the first contact with the app, or when strategizing complex tactical setups.

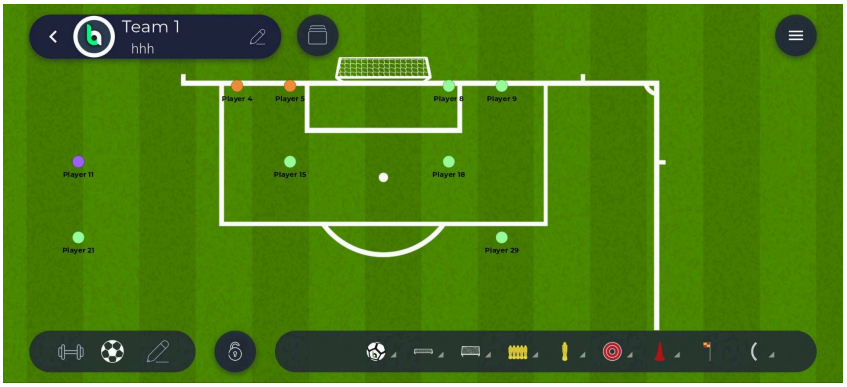

Figure 5: bCoach Tactical Pitch UI

e. ***Data and content requirement***

The app requires the user to input basic team data during account sign-up, particularly the number of players, which seems to affect how certain features function. Once inside, the app relies heavily on user inputs for tracking match events and storing tactical information. This makes it somewhat content-heavy, as coaches need to build and maintain a detailed setup of their team's activities and performance manually.

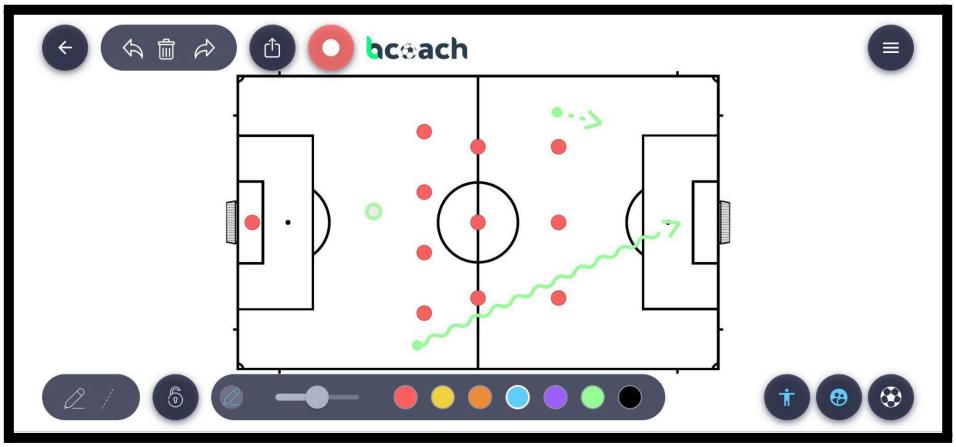

Figure 6: bCoach Tactical Whiteboard UI

CoachBetter

1. ***Analysis***

a. ***App functions***

CoachBetter advertises itself as "the most efficient app for football coaches and clubs". Its major standout is training planning, with a really professional library of over 700 exercises and knowledge articles for session design, with complexity levels for all types of coaches, from complete beginners to professionals.

It also includes tools to create match tactics, evaluate players, and track development via individual player profiles with performance data. Another distinct feature, apart from in-app messaging for players, staff, and parents, is the integrated fee management functionality. Overall, it is the most complete coaching application we've come across in our analysis.

b. ***User flows***
On first login, CoachBetter requires users to select a role (Coach, Player or Parent) and, for coaches, complete a three-question quiz to establish a personality profile. Coaches then enter their club's name and age group, but might encounter a considerable number of "Did you know?" pop-ups and mini-tutorials before accessing main functions. Players must be invited by a coach via a custom link and cannot view lineups made by the coach, limiting end-to-end transparency.

c. ***UI and style***
CoachBetter adopts a minimalistic palette, with one primary color with accents and neutral text, but suffers from occasional alignment glitches and inconsistent iconography. Interactions and swipe motions are generally smooth, yet explanatory overlays can feel intrusive and delay task completion.

d. ***Type of use***
Its use feels lighter and more suitable for multi-session engagement, with coaches being able to return spontaneously to plan sessions, update lineups, and review statistics. The player app focuses on availability and limited messaging, which also suits the multi-session type.

e. ***Data and content requirement***
All content (player profiles, training activities, match events) is manually entered by coaches via form fields or drag-and-drop. There is integration with cloud authentication and it deploys an account system. It also has the functionality of synchronizing app events with the user's Google Calendar. While statistics can be logged from match highlights, they often fail to populate correctly in player profiles.

2. ***User Reviews & Feedback***
Despite the app's very complete and professional approach, boasting partnerships with a multitude of first-division clubs around the world, its reviews by users are actually mostly negative. For the most part, there is a staggering number of complaints regarding data sync and how the app feels unintuitive.

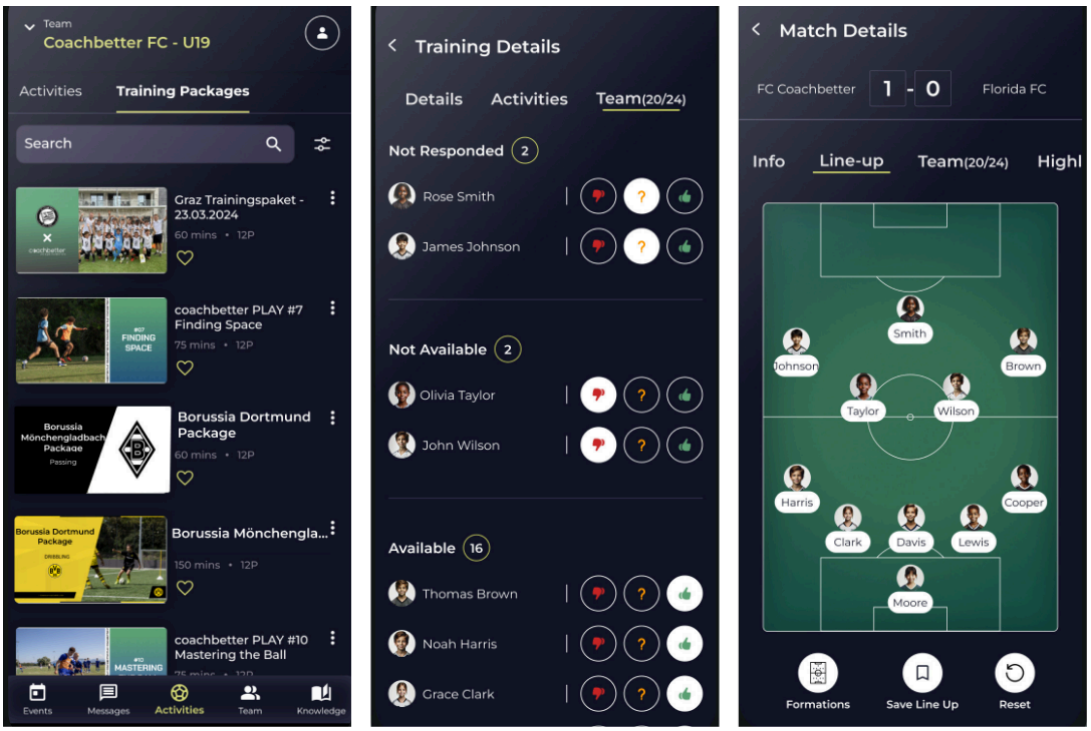

Figure 7 CoachBetter UI

Across these examples several recurrent themes arise. First, many existing apps are either too simplistic in scope or too complex in execution. Second, stability, clarity and learnability are frequently more important to grassroots users than advanced analytics. Third, features that require continuous or real-time input are only useful if the app performs well on low-end hardware and unstable networks. These conclusions guide Coach Canvas toward a middle ground, with focused core workflows, pragmatic technical choices and forgiving interactions that work on modest devices.

Critical analysis and implications

The competitor landscape, therefore, highlights three strategic implications. Firstly, to prioritise a small number of high-value use cases rather than attempting a broad feature set from day one. Secondly, to invest in predictable, discoverable UI patterns that are resilient to user error. And third, to account for real world constraints of grassroots clubs: (older devices, limited data, volunteers with limited training time and social dynamics that complicate ownership and moderation).

3.2. Technologies

Platform and software choices

The technical approach for Coach Canvas prioritises rapid cross-platform delivery, low connectivity adaptability and discrete resource demand for optimal suitability in lower end devices. Despite originally leaning towards the Google-made Flutter (based in C+) for its simplicity and front-end usability, React Native was selected as the primary framework because of the familiarity with the course content and guidance available.

This choice reduces development and testing effort and reflects the project constraint to produce a demonstrable mobile prototype within an academic term.

Firebase was chosen as the backend for authentication, realtime synchronization and storage. Firebase Authentication simplifies social and Google sign-in flows that coaches expect, and Firestore provides a scalable, document-oriented model that maps well to the club/member/lineup data structures needed for the app. Realtime capabilities and offline persistence in Firestore also help make availability status updates and lineup edits feel immediate while tolerating poor connectivity.

Distribution and promotion

Distribution followed the standard mobile route, with APK exports and built-in Android emulations during development, and an expectation to eventually submit to Google Play for a wider reach. Early promotion and participant recruitment has been envisaged through local club outreach, student networks, Reddit and coaching communities, as well as in-person testing at training sessions. The promotional approach focuses on targeted, low-cost channels to reach grassroots managers and to gain practical testing partners rather than pursuing broad marketing in the early stages.

Research tools and methods

Methodologically, the project combines fieldwork, paper prototyping and iterative digital prototyping. Tools included Figma for wireframes and high fidelity designs, React Native for the working prototype, and simple analytics instrumentation to capture in-app events during tests.

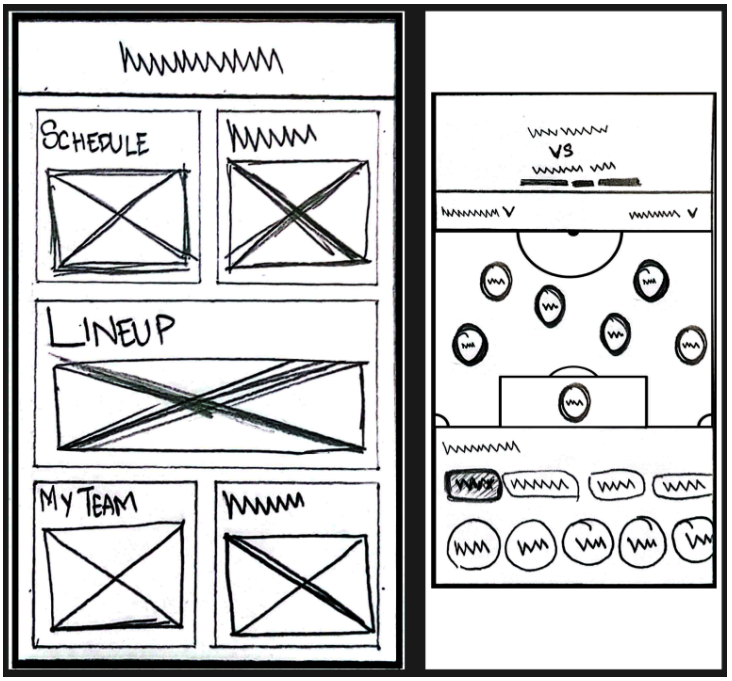

Figure 8: Early paper prototype design

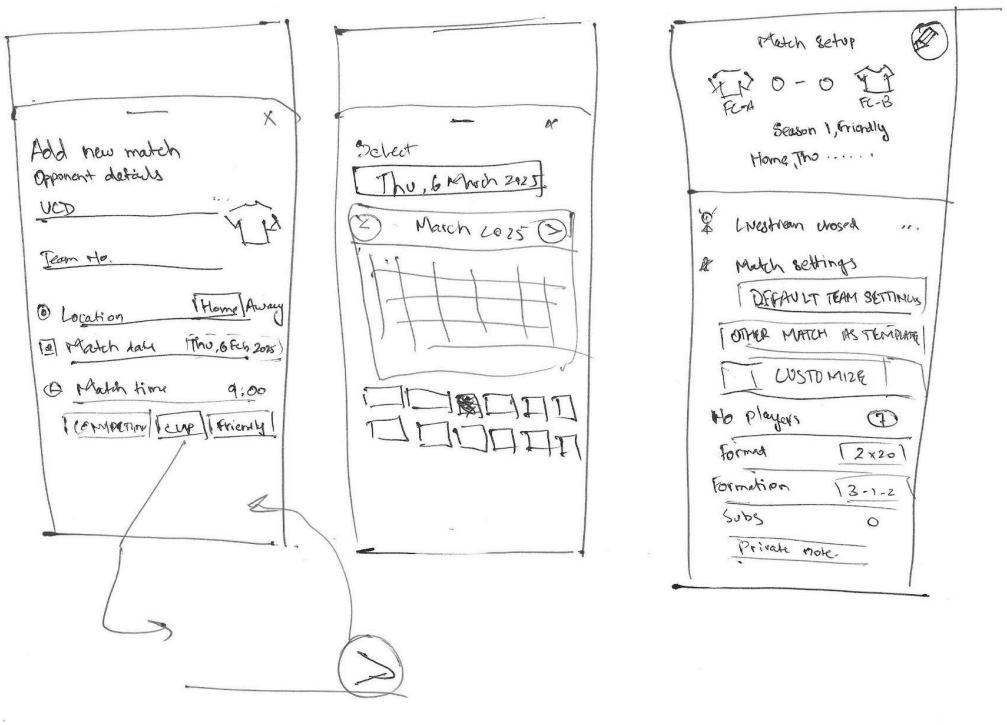

Figure 9: Early paper prototype interaction design

Recruitment and qualitative testing relied on local clubs and casual networking, reflecting the pragmatic needs of real world coaches. Ethical considerations, particularly around minors, were managed by anonymising data and limiting sensitive fields, matching the low data footprint expected in the app.

3.3. Definition Of Design Requirements

Audience and rationale

Coach Canvas targets managers and players in grassroots and semi-professional settings. These users share a requirement for quick, reliable tools that reduce coordination load and communicate tactics clearly. From the competitor analysis and field observations the following design imperatives emerged:

- ***Simplicity with purpose***
  The app must foreground four primary workflows (Schedule, Squad, Tactics and Match Mode). These functions map directly to the most frequent pain points identified by users and allow narrow, measurable goals for usability.

- ***Mobile first and resilient***
  Interactions must be optimised for phones used on the touchline and on the move, and for devices with limited processing power. This implies larger tap targets, minimal animation, and a load-minimising approach for critical flows.

- ***Fast sharing***
  Coaches value the ability to share a tactical setup with one tap. Coach Canvas

must deliver a simple export usability that produces a clean, legible field view of the selected tactics.

- ***Clear availability and quick decisions***
  Availability is the recurring single biggest friction. The Squad view must surface availability at a glance and the Upcoming Match view must allow the manager to generate or easily create a starting eleven based on availability and simple position matching.

- ***Simple club membership model***
  Inspired by the Clan model, clubs exist independently of a single manager and support simple, auto-generated, reusable join codes. The design favours multiple managers where possible to reduce single point owner problems and supports recovery flows if a manager leaves.

- ***Privacy and data minimisation***
  Given the presence of minors in some target teams, the app defaults to private player profiles, collects minimal sensitive data and offers clear consent flows for contact information. This choice reduces risk and aligns with grassroots expectations.

- ***Microcopy and onboarding***
  Small, contextual hints and a brief in-app demo for key interactions reduce the learning curve identified in competitor apps. The onboarding balances brevity with clarity, surfacing only essential first actions.

Specific feature decisions informed by competitors

From Coach Tactic Board we adopt an emphasis on tactical clarity and a visual pitch canvas, but we reject heavy animation and frame-based drill editors that increase complexity. From Coach Amigo we learn to avoid crowded dashboards and instead present a focused home hub. From bCoach and CoachBetter we take lessons about real-time event tracking and training libraries, but treat those as lower priority features to be rolled out after the core flows are stable. Non-domain inspirations inform affordances: Trello-style cards for quick reordering of substitutes, Figma-style selection handles in the lineup editor, and Duolingo-inspired guided first-time hints.

Trade-offs and constraints

The design choices require trade offs. Limiting roles to Manager and Player simplifies the UX but increases the need for careful ownership and recovery policies. Prioritising offline resilience and low device load constrains the fidelity of animations and the complexity of in-app editing tools. Choosing React Native and Firebase accelerates development but imposes vendor lock-in for backend services and requires deliberate attention to Firestore billing and security rules during scaling.

3.4. Conclusion

The background research validated the original ambition for Coach Canvas: there is a tangible need for a lightweight, tightly focused tool that reduces the administrative and communicative friction faced by grassroots managers and players. Competitor analysis exposed common failures to balance depth with usability, and non-domain inspirations offered pragmatic interaction patterns that transfer well to the coaching context. The technological choices support rapid prototyping and the delivery of mobile-first, resilient experiences. The derived design requirements are intentionally conservative: core flows must be robust, discoverable and fast, with richer features deferred until the basic experience is proven reliable in the field. These conclusions provided a clear specification for the prototyping and testing phases described in subsequent chapters.

4. METHODOLOGY & APPROACH

4.1. Introduction

This chapter outlines the practical processes through which Coach Canvas was designed, developed, and iteratively refined. The chapter therefore focuses on two complementary fields (the design methodology used to define the structure and behaviour of the interface, and the technical approach adopted during implementation).

The development process began once all the analog prototyping phases had been completed, ultimately progressing from early ideation and low-fidelity sketches to higher-fidelity wireframes, interactive prototypes, and the creation of the final user interface design system. The chapter will include the choice of frameworks and technologies, state management strategies, data structuring decisions, and the integration of features such as authentication, scheduling, club management, and lineup creation. It will also highlight technical challenges encountered during the process, the solutions implemented, and the reasoning that informed these decisions.

4.2. Design

Since early on in the conceptualisation of Coach Canvas, the intent was for the visual design to complement the usability, instead of being a limiting factor. Therefore, accessibility and familiarity was prioritised, taking into consideration that target users would already be familiar with the general visual language and affordances of football.

Analysis of competitors and similar apps allowed us to identify common patterns that communicate concepts such as tactics, availability, or goals/assists.

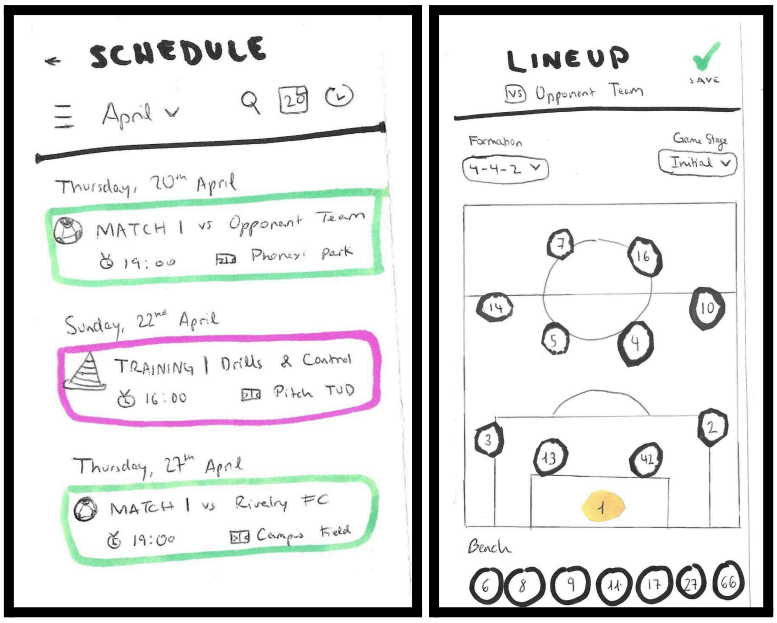

Figure 10: Paper prototype Schedule and Lineup viewer design

User Interface

1. Colour Scheme

The app adopts a dark-mode-first palette rooted in deep, muted green tones with a complementary blue as the primary highlight accent. Dark backgrounds are perfect for colour contrast, and improve battery life on most devices.
Greens were chosen for their semantic alignment to pitch, tactics and sport, while blue was reserved for interactive elements to create a clear visual hierarchy. All colour combinations were evaluated against minimum contrast guidance to ensure readability for primary information and controls, and alternative high-contrast styles were prepared for accessibility testing.

Figure 11: Coach Canvas branding and colour scheme

2. Typography

Inter was selected as the system typeface because it is widely available, developer-friendly and performs consistently across all Android devices. Its legibility is still very well rated at small sizes, which is important for reading names and numbers on the lineup view. A small typographic scale with a clear hierarchy was also defined for consistency.

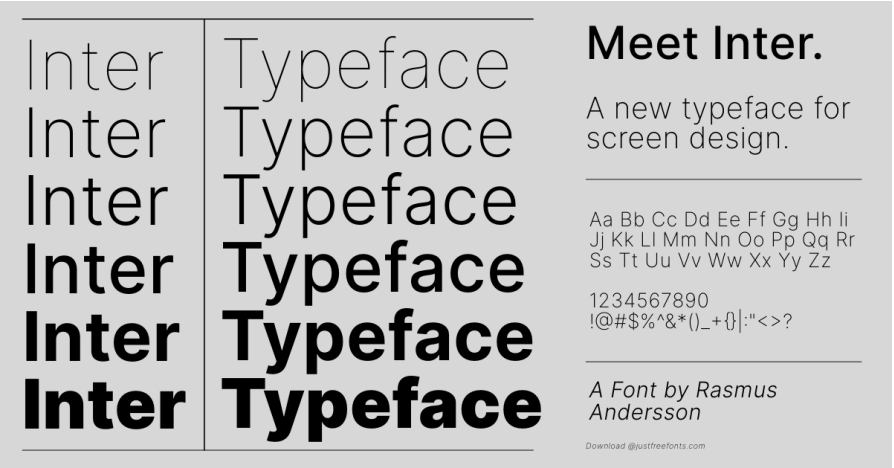

Figure 12: Inter typography family

3. Shapes, iconography and imagery

We opted for rounded corners for cards, toasts, buttons and panels to convey a friendly, approachable aesthetic. Vector-based iconography was preferred to photographic imagery so that the interface and the size of the application remains light-weight, consistent and scalable.

Icons are established to be simple, clear and legible. Some of the iconography was obtained from the repository site Iconify, while others were self-produced to match the chosen aesthetic. Where domain-specific meaning was required (for example a cone to signify "training" or a boot to signify "assist"), established relationships observed in competitors and general football media were used, making it easy for users to recognise meanings immediately. Player representation is mostly card-based, with compact cards displaying initials (as placeholder)/photograph (if set), squad number, position tag and a small availability indicator. These cards are highly reusable across the Squad, Bench and Lineup views.

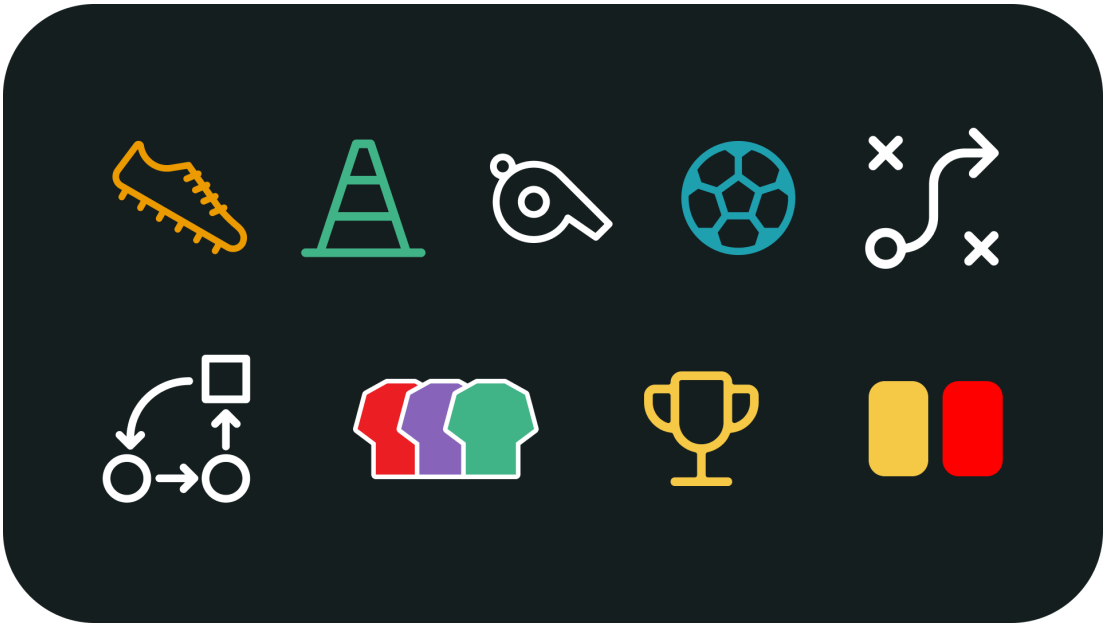

Figure 13: Visual language and iconography for Coach Canvas

4. Microcopy and affordances
Microcopy plays a central role in the UI, providing context-sensitive hints (for example, when choosing how to create a lineup) and preventing accidental actions (for example confirming lineup sharing). Care was taken to keep them short and concise.

5. Feedback, animation and performance
Micro-interactions deliver necessary feedback while keeping animations minimal in duration and complexity because of device diversity. Examples include a short outer highlight when a player is selected, a top-emerging toast when a player is subbed on, and a toast confirmation when a lineup is shared, among many other observable UX refinements. These are optimised so the interface feels responsive on lower-end Android devices. Loading states and error states are also, with clear messages and actions to take.

6. Accessibility considerations
Colour contrast, large tap targets and an emphasis on text labels reduce reliance on colour alone. Icons are accompanied by accessible labels, and the interface respects system font size settings.

Layout

1. Home screen and information hierarchy
The Home screen functions as the main hub, surfacing the primary areas with prominent tiles: Schedule, Squad, Tactics, Match History and Upcoming Match (if available). The sections are visually distinct, and we aimed to limit navigation depth ensuring users can reach core tasks in as few taps as possible.

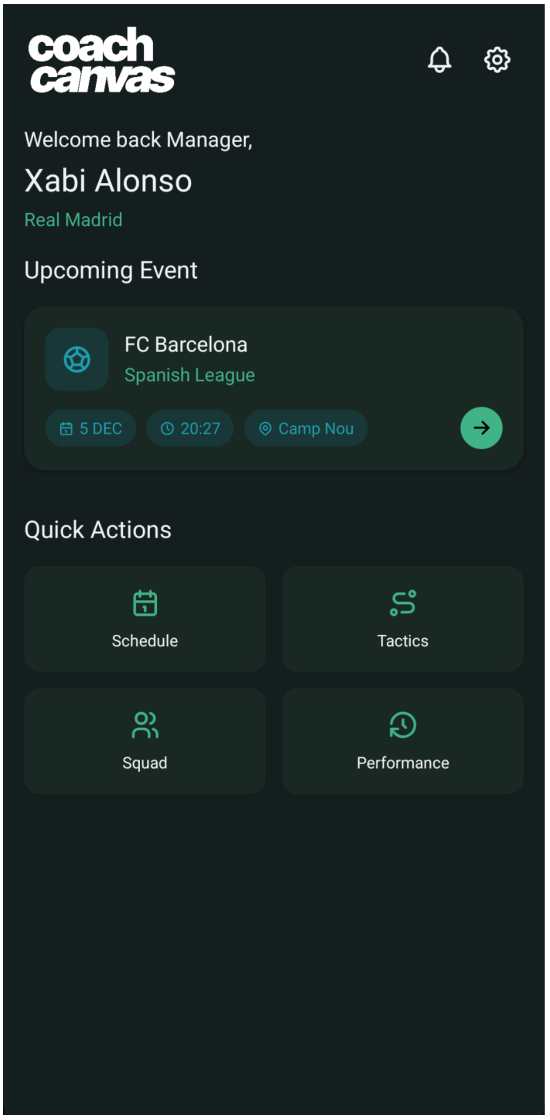

Figure 14: Home screen design

2. Spacing

A modular grid was defined in the Home screen to ensure consistent composition and visibility. In general, spacing rules prioritise readable density. Taking player cards within the lineup view as an example, they are compact but not cramped.

3. Tactical canvas and pitch layout

The pitch canvas uses a responsive coordinate grid so the different formations read well on different aspect ratios. Player cards show number, name and an abbreviated position label. The bottom toolbar lets managers switch formation presets and access the bench.

Figure 15: Lineup canvas screen

4. Event and schedule presentation

Schedule screens offer a compact vertical list for easy navigation, with three filter tags including All Events, Matches or Trainings. Event cards use colour-coding to distinguish a match from a training session, and include a compact availability overview badge. Creating or editing an event is done within the event modal to preserve context, with focused fields and clear action buttons.

5. Responsive considerations

Layouts were optimised for performance and low-bandwidth operation, generally transmitting simple text data. Heavy assets are avoided and all icons are vector-based. The layout margins and visual densities were tuned so that key elements remain legible even with system font scaling.

Navigation

1. Core navigation model
Given the straight-forwardness of the application's nature, we opted for not including a static navbar at the bottom. In turn, the most usual "deep" actions, such as creating a lineup or an event in the Schedule, are prominently surfaced with thumb-reachable buttons in the relevant screens, rather than nested inside menus.

2. Role-aware entry points
The navigation is role-aware, with managers being able to see Edit and Create controls within the Squad and Upcoming Match screens, while players see read-only tactical views and availability toggles. This reduces clutter and ensures users are not confronted with irrelevant controls. A unified Home screen layout remains common to both roles to maintain conceptual consistency.

3. Error handling and recovery paths
Navigation errors are handled with clear, recoverable flows, such as unsaved changes prompting a confirmation, and onboarding conflicts providing visible guidance. This keeps the app tolerant of the interruptions and social friction common in football situational settings.

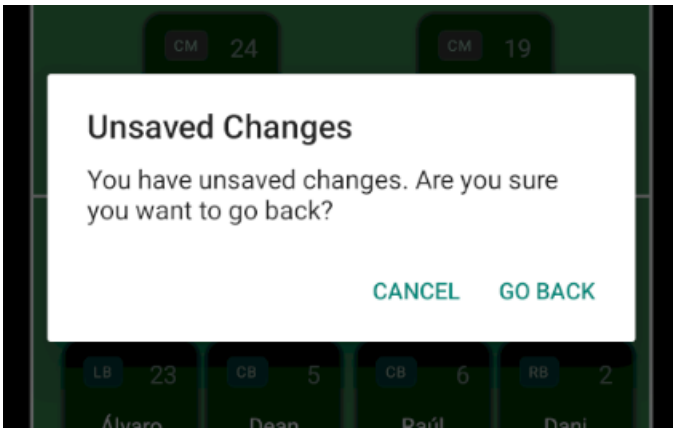

Figure 16: Unsaved changes toast

4.3. Code

The implementation of CoachCanvas is centred around a feature-based, TypeScript-first codebase. All source code lives under `/src/`, which is split into domains such as `components`, `screens`, `services`, `navigation`, `contexts`, `hooks`, `types`, and `utils`. With this structure, each folder represents a clear responsibility, which keeps the mental model of the project small even as the codebase grows. The service layer is the heart of the application. These are wrappers around Firebase SDKs which encode the domain rules of the app (what it means to create a club, approve a join request, or record a match action).

Components and screens never talk directly to Firestore or Auth, instead they call services. This separation allows the UI to evolve (or replace what's needed) without

rewriting logic. All Firestore documents have matching interfaces in `src/types`, and service methods are fully typed for parameters and return values, which prevents a large class of runtime errors. Node plugins ESLint and Prettier enforce a consistent style, while path aliases and shared utility functions (`src/utils`) reduce duplication.

clubService.ts

```ts
// clubService.ts
class ClubService {
  private clubsCollection = firestore().collection('clubs');

  async createClub(
    clientRequestId: string,
    ownerUserId: string,
    data: ClubFormData,
  ): Promise<Club> {
    // Idempotency: avoid duplicate clubs on retries or double taps
    const existing = await this.clubsCollection
      .where('clientRequestId', '==', clientRequestId)
      .limit(1)
      .get();

    if (!existing.empty) {
      return existing.docs[0].data() as Club;
    }

    return firestore().runTransaction(async transaction => {
      const clubRef = this.clubsCollection.doc();
      const memberRef = clubRef.collection('members').doc(ownerUserId);

      transaction.set(clubRef, { ...data, clientRequestId });
      transaction.set(memberRef, { role: 'OWNER', createdAt: Date.now() });

      return { id: clubRef.id, ...data } as Club;
    });
  }
}
```

Figure 17: Example service implementation in `clubService`

This snippet shows the typical pattern (encapsulated Firestore access, duplicate handling, and transaction usage in a single method.

Code workflow

The development workflow for Coach Canvas followed a "service-first, then UI" methodology. For each new feature, the general flow we opted for was:

1. Define the data structures and types in `src/types` (such as `club`, `event`, `playerStats`).
2. Implement or extend a corresponding service (such as `clubService`, `eventService`, `playerStatsService`) with methods that expand communication.
3. Build or adapt screens and components that call these services, connecting them through React hooks and context.
4. Integrate the new screens into the navigation graph.

5. Run manual QC tests on Android (first only via emulator, as soon as it was feasible we opted for exported APKs in our devices). These processes helped refine error handling, loading states, and UX.

On the runtime side, the code follows a unidirectional data flow (a user interacts with a screen → the screen triggers an event handler → the handler calls a service → the service reads/writes Firestore or Auth → the result updates React state → the UI re-renders. Firestore listeners are used for persistent data (events, availability, notifications), while "one-shot" reads are used for actions like creating a club or submitting a join request (that are generally only utilised once by the user).

Error handling and loading are part of the workflow by design, as each screen keeps a local `isLoading` state for its own operations. Services always throw typed `Error` objects with user-friendly messages, and the components then display these via a Toast or modal, depending on severity.

EventListScreen.tsx

```tsx
// EventListScreen.tsx
const EventListScreen: React.FC = () => {
  const { currentClubId } = useClubContext();
  const [events, setEvents] = useState<EventSummary[]>([]);
  const [isLoading, setIsLoading] = useState(true);

  useEffect(() => {
    const unsubscribe = eventService.subscribeToEvents(
      currentClubId,
      nextEvents => {
        setEvents(nextEvents);
        setIsLoading(false);
      },
      error => {
        Toast.show({ message: error.message, type: 'error' });
        setIsLoading(false);
      },
    );

    return unsubscribe;
  }, [currentClubId]);

  // ...render list or skeletons
};
```

Figure 18: Screen–service interaction in an event screen

This pattern, subscribing through a service and showcasing errors via a UI toast, is repeated across major data-driven screens.

Architecture

Architecturally, the app combines Service-Oriented Architecture (SOA) with component-based UI and React Context for global state. The goal was to make each layer responsible for only one kind of concern:

- ***Presentation layer:*** React Native screens and components in `src/screens` and `src/components`.

- ***State and orchestration:*** Context providers (mainly `AuthContext`) and custom hooks (`useEvents`, `useSubstitutionManager`).
- ***Domain logic and data access:*** Service classes in `src/services`.
- ***Infrastructure:*** Firebase (Auth, Firestore, Cloud Messaging, Storage) and Android native configuration.

The services are grouped by domain rather than by technology. For example, `playerStatsService` owns the logic to aggregate goals, assists, and appearances across matches (via Game Mode), while `matchResultService` is responsible for storing outcomes and scores. This mirrors the mental model of the user (club, player, event, lineup, stats) instead of the physical model of the database. Data is stored in Firestore using a nested collection structure (`users`, `clubs` including subcollections `members`, `joinRequests`, `events`, `lineups`, `players`, and `notifications`. The code formalizes this structure in React, and transactional operations (club creation, squad number reservation, join request approval) are always implemented as Firestore transactions to maintain consistency.

substitutionUtils.ts

```ts
// substitutionUtils.ts
export const POSITION_COMPATIBILITY: Record<PositionCode, PositionCode[]> = {
  GK: ['GK'],
  CB: ['CB', 'LB', 'RB'],
  LB: ['LB', 'CB'],
  RB: ['RB', 'CB'],
  CDM: ['CDM', 'CM', 'CB'],
  CM: ['CM', 'CDM', 'CAM', 'LM', 'RM'],
  CAM: ['CAM', 'CM', 'ST', 'CF'],
  ST: ['ST', 'CF', 'CAM'],
  CF: ['CF', 'ST', 'CAM'],
  // ...
};

export function canSubstitute(
  from: PositionCode,
  to: PositionCode,
): boolean {
  return POSITION_COMPATIBILITY[from]?.includes(to) ?? false;
}
```

Figure 19: Example of the position compatibility architecture

This snippet illustrates how tactical constraints are encoded, rather than scattered across front-end logic, keeping substitution rules maintainable across versions and easily testable.

Functionality

From a code perspective, each functional area of the app corresponds to one or more service modules and a set of screens:

- Authentication and onboarding are implemented by `authService` and `userService` plus screens in `src/screens/auth` and `src/screens/onboarding`. Email/password and Google sign-in are both

handled through a common logic to display errors to the user. Onboarding flows are driven by guidance screens that pass typed form data between steps until submission to the service layer.

- Club and member management is encapsulated in `clubService`. It generates join codes, handles duplicates (`clientRequestId`), manages members and `joinRequests` subcollections, and provides utilities for squad number reservation. These methods are used by manager screens to create clubs, approve players, and update club details.

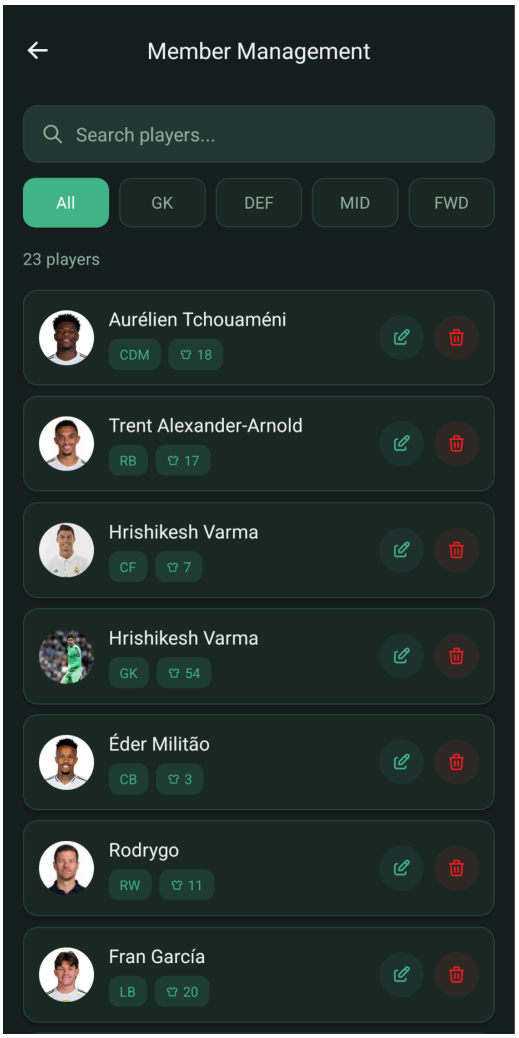

Figure 20: Member management

- Events and availability are handled by `eventService` and `availabilityService`. Event creation methods accept details such as date, time, competition (which you can save as presets with their own specifications), location, and availability deadlines, while listeners expose event lists and per-event availability in real time. Manager and player screens subscribe to these streams to keep the UI synchronized.

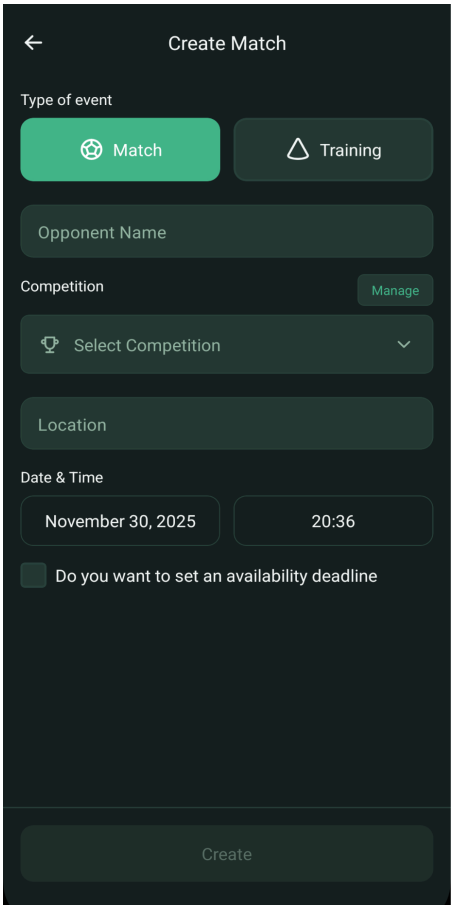

Figure 21: Event creation screen

- Lineups and substitutions rely on `lineupService` and the `useSubstitutionManager` hook. Lineups are saved as documents containing format, formation, and a map of positions to player IDs. The hook manages local state for the working lineup, applies compatibility rules, and exposes safe methods like `swapStarterAndBench` to the UI.

- Player management and stats uses `playerService`, `playerStatsService`, `matchResultService`, and `matchActionService`. These services operate together (match actions are written during or after games, results are stored at the match level, and stats service adds this data per player.) Screens in `src/screens/shared` (such as player profile, match history detail) render the aggregated view.

To keep functionality coherent, each screen imports only the services and hooks it needs for its domain. Navigation simply determines which combination of screens is active for a given user role, but the underlying services remain the same.

availabilityService.ts

```ts
// availabilityService.ts
async function setAvailability(
  clubId: string,
  eventId: string,
  userId: string,
  status: AvailabilityStatus,
  note?: string,
): Promise<void> {
  const docRef = firestore()
    .collection('clubs').doc(clubId)
    .collection('events').doc(eventId)
    .collection('availability').doc(userId);

  await docRef.set(
    {
      status,
      note: note ?? null,
      updatedAt: firestore.FieldValue.serverTimestamp(),
    },
    { merge: true },
  );
}
```

Figure 22: Example of status update functionality for availability

This method backs the UI toggle where players mark themselves as available, unavailable, or maybe for a given event.

Other code notes

Several implementation details improved maintainability and performance without being visible to end users:

Firstly, the project uses batching and Map-based lookups for Firestore queries that involve many player IDs. It's worth noting that Firestore limits the size of queries to 10 IDs, so services like `lineupService` and `clubService` slice input arrays into chunks of 10 items, then perform multiple queries, and merge the results into `Map<string, Player>` objects. This makes rendering squads and lineups with large teams both feasible and fast.

Secondly, skeleton loaders were introduced as dedicated components (`PlayerCardSkeleton`, `StatCardSkeleton`, `MatchHistoryDetailSkeleton`, `NotificationSkeleton`). Rather than showing spinners or blank screens, data-heavy views render these placeholders while data streams in from Firestore listeners. This is a small amount of extra code but greatly improves perceived speed.
Third, the codebase includes the foundation for automated tests via the Node plugin Jest. The initial tests focus on rendering the root `App` component and verifying that core navigation functions correctly.

Figure 23: Example Jest configuration

This configuration prepares the baseline for future testing and upgrading around services, hooks, and UI components.

General relevant observations

A few additional coding choices are worth noting even though they do not fit strictly into the previous sub-sections.

- ***Security and rules:*** Firestore and Storage security rules (such as in `firestore.rules` and `storage.rules`) were written to align with the app's role-based model. Managers can modify club and event data for their clubs, while players can only modify their own availability and profile information. The service layer is implemented under the assumption that these rules are enforced server-side.
- ***Native integration and configuration:*** React Native Firebase modules require native configuration (`google-services.json` on Android). The project's native folders (`android/`) contain the necessary Gradle settings for Firebase, push notifications, and asset linking. These are mostly boilerplate but are essential for features like Cloud Messaging to work reliably on devices.
- ***Scalability:*** The architecture is intentionally prepared for future backends or advanced features. Because all the Firebase logic is isolated in services, it is feasible to replace Firestore with another API layer or to introduce serverless functions without rewriting UI components. The same way, the baseline code in `src/types` will help when adding advanced analytics, offline caching, or even some sort of future AI-assisted lineup suggestions.

Together, these methodological and architectural choices result in a codebase that is highly functional for the current scope of Coach Canvas, as well as structured to evolve and remain maintainable as new requirements emerge.

Use of AI-assisted development tools

During development, generative LLM tools such as Claude and the code editor Cursor were incorporated selectively to improve workflow efficiency. These tools were used primarily to streamline repetitive tasks (such as assembling components, identifying minor bugs, and generating boilerplate), to explore alternative implementation

approaches (curating Stack Overflow queries, for example), and to validate architectural decisions.

The use of these tools did not replace core development work, only helping to speed up experimentation and offering explanations that contributed to a deeper understanding of certain plugin features or technologies, with critical design and implementation decisions remaining led by our team.

5. TESTING & ANALYSIS

5.1. Introduction

This chapter documents the testing programme carried out during the design and development of Coach Canvas, and it analyses the results and the consequent design decisions. The testing activity combined informal, contextual fieldwork with progressively higher fidelity and depth prototypes and coded builds. The purpose of the testing programme was to first, validate the core value proposition; that a compact, mobile-first hub with the Schedule, Squad and Match userflows would be beneficial for amateur and semi-professional managers and players. And secondly, to discover where the interface, interaction intricacies and data flows needed refinement in order to be robust on the kinds of devices and in the contexts most typical of grassroots football.

We will first describe the overall methodology and technologies used in prototyping and testing, then present detailed results and sessions from each test cycle, illustrating tasks, participant profiles, observed behaviours, quoted material and emergent issues. The next section aggregates findings across sessions and translates them into concrete design decisions that were implemented during the development period..

5.2. Methodology & Test Environment

Approach

The test programme used iterative, mixed-method evaluation. Where possible, tests were contextual and deliberately concise, reflecting real coaching constraints (busy training sessions, short breaks and informal sideline situations). Methods included structured task-based testing, with think-aloud prompts in informal sessions and open-ended questions. Tests progressed from low fidelity paper sketches through interactive Figma prototypes to coded builds so that each round could validate the next set of assumptions, until reaching nearly the last state of the application.

Participants and recruitment

Participants were recruited from local clubs and the Technological University Dublin community. Recruitment focused on the target population (managers and coaches of ages 16–23, academies and serious amateur senior teams belonging to the Football Irish Association and the Spanish Prodepor Federation, and players affiliated to those teams. Across the programme, 17 people participated in formal and informal tests (7 coaches/assistants), and 10 players ranging from U17s to over 24. This spread allowed the project to capture both managerial concerns and player expectations.

Prototypes and technologies

Prototypes used in tests were:

- Paper sketches and modular cut-outs for initial interaction exploration.

- Figma low/mid-fidelity interactive screens to test flow and affordances.
- A small Flutter-coded MVP concentrating on lineup creation for rapid, touchline-style validation.
- A React Native prototype integrated with Firebase for the most complete tests, covering club join, events, saved lineups and basic persistence.

Testing conditions and metrics

Sessions were conducted in naturalistic settings where possible, such as training breaks, sideline after matches and clubrooms. For each test we defined a set of primary tasks and secondary questions. Common primary tasks included locating the next training in the schedule, creating a lineup, swapping two players, accepting join codes and checking player availability.

Metrics collected were primarily qualitative, such as task completion success, common errors, timing where possible and most importantly, participant impressions. No formal large-scale quantitative survey was carried out, but recurring issues and themes were counted across sessions to analyze prevalence, which proved efficient in the end.

5.3. Testing Sessions In Detail

Session 1: Figma prototype | Willows FC U17s coaches
Date: 25 July 2025
Participants: 2 (Head coach and assistant)

Purpose and tasks
This early field test used Figma interactive screens that mirrored the Home, Schedule, Squad and Upcoming Match flows. Sessions were short, carried out during a training break from the team, and aimed to surface immediate expectations for navigation, lineup creation and availability management.

Questions prepared
- How would you get to the next training on this screen?
- If I asked you to pick the starting eleven for Saturday, where would you do that?
- How would you tell who is available or not?
- If I wanted to swap two players for the match, what would you try first?
- Is there anything you feel is missing right now?
- Could you get a lineup ready from scratch during a 10 minute half time?
- What would make you trust the app enough to stop using WhatsApp for availability?
- Do you want formations presets or a “build your own” each time?
- Any quick polish that you would recommend?

Key tasks
- Find the next training session.
- Prepare the starting eleven for a match.
- Identify which players are available.
- Swap two players in a lineup.

Findings
Both participants appreciated the concept of a single hub and found the Home screen conceptually correct. Important issues included:

- Both coaches attempted drag-and-drop for player movement, but the prototype used tap-to-swap and participants commented that drag would feel more natural "on a tablet" while tap was acceptable on phones with smaller sized screens.
- Coaches requested a prominent availability toggle and quick glance attendance summaries.
- The coaches suggested filters such as left-footed or height-based filters for quick tactical decisions within the Squad view.
- One coach nearly shared the lineup prematurely and suggested some sort of confirmation step before sharing to the squad.

Takeaways and influence
The session confirmed that a compact home hub makes sense to our target users. It also established a clear interaction conflict, with users expecting drag-and-drop interaction, but device variability and developing concerns mean the app needs a robust alternative. Design decisions taken: make tap-to-swap the default, provide an optional drag mode for devices that support it, and add explicit lock-before-share affordances. Availability indicators were promoted to the Squad view as a high-priority element.

Session 2: Coded MVP in Flutter | McKelvey FC U18s C
Date: 16 August 2025
Participants: 3 (coach + 2 players)

Purpose and tasks
A functional, coded MVP focusing exclusively on lineup creation usability was tested on the sideline after a match.

Questions prepared
- How would you move players around?
- How would you swap two players if you wanted to change their positions?
- If you finished the lineup, how would you share it with the team?
- Would you want to lock it so nobody can change it after you share?
- Anything about the way players are shown that you like or dislike?

Key tasks
- Place players into positions on the pitch.
- Swap two players.
- Share the finalized lineup as an export.

Findings
The MVP was praised for speed, clarity and directness. Key observations included:

- Bench visibility: participants requested a larger bench area and less scrolling so substitutes are as visible as possible.

- Sharing expectation: coaches explicitly asked for a quick image-file export suitable for posting to group chats or social media.
- Position labelling: players wanted position labels visible (left wing, defensive mid) rather than just name and/or number.

Takeaways and influence
The MVP validated the core claim that a simple, fast lineup tool adds immediate value in the matchday. From this session, we learned to implement a large bench UI in future upgrades, and to add compact position labels to player cards.

Session 3: React Native prototype | Willows FC assistant coach
Date: 8 September 2025
Participants: 1 (assistant coach)

Purpose and tasks
This test evaluated the onboarding flow, Google sign-in, club creation/join, and basic schedule and lineup interactions.

Questions prepared:
- Could you try to create an account for yourself as a coach?
- How would you prefer to invite the players into the app?
- How easy was it to set up your account?
- When you land on the main menu, what's the first thing you'd tap?
- How would you create a training session for next week?
- If you check the lineup option, how would you put a player into a position?
- Would you want your players to see everything the same way?
- Anything that feels like it takes too long or too many taps?

Key tasks
- Create an account and invite players.
- Create a training event.
- Place a player in a lineup.

Findings
The assistant appreciated Google sign-in and the central schedule view. Concerns included:

- Onboarding friction: the assistant wondered whether players should be required to create an account to merely check if they are selected. The idea of a lighter "player mode" that didn't involve accounts was raised.
- Filters and bench controls: clearer position filters were requested.
- Lineup movement: free movement modes and a clear definition of specific positions, such as right-back (RB) as opposed to right-wingback (RWB) were areas of confusion.

Takeaways and influence
The session reinforced the importance of minimizing sign-up friction and brought up the idea of an account-less usage for the app for players who wanted as little involvement as

possible. However, this was eventually deemed technically unfeasible, opting instead to clarify and inform users of why an account is required.
Therefore, our development conclusions were to keep full sign-up but improve onboarding microcopy, and to implement positional filters and never assume knowledge from users, always keeping the app accessible to less tactically literate users.

Session 4: App prototype 1.1 | TUD players
Date: 2 October
Participants: 3 (university players)

Purpose and tasks
This testing session validated club joining, events, and saving lineups. Participants tested in a relaxed manner around a campus pitch.

Key tasks
- Join a club via code.
- Set availability.
- Create and save lineups.

Findings
Participants tried the drag-and-drop interaction as a first instinct, but felt comfortable once told that player swapping worked via tapping. One user expressed willingness to pay for advanced features, such as player card customisation within the publicly shared lineup. Others flagged small UX annoyances such as unclear lineup saving flows, in this occasion simply due to lack of technical implementation.

Takeaways and influence
The session reinforced earlier conclusions regarding lineup UX flows (make saving explicit with confirmation toasts) and surfaced pricing and premium option ideas that were disregarded later (or left aside until future versions).

Session 5: Group testing | Antzarak CF (Spain)
Date: 13 October
Participants: 7 (5 players, 2 coaches)

Purpose and tasks
This larger group test was an incredible opportunity to test multi-user dynamics, club join behaviour, manager tools for saved lineups, event creation and sharing. Tasks were split between managers and players.

Key tasks for managers
- Create a club and generate a join code.
- Create a default lineup and save it.
- Create a match event and save a match-specific lineup.

Questions prepared for managers:
- When building the lineup, how intuitive did the screens feel for placing players into positions?

- Did the saved lineup retrieval feel discoverable in case you want to reuse a formation?
- Would you feel comfortable using this flow before a game?
- If you could change something, what would it be?

Key tasks for players

- Join via code and set up a profile.
- View the scheduled match and read manager notes.

Questions prepared for players:

- When you opened the match, did you check the notes from the manager?
- Would you be interested in viewing the stats for other players in the squad, or just yours?
- If you could change something, what would it be?

Findings

This session surfaced high value feedback and some usability defects:

- Statistics drive competition: the possibility to track match stats and be able to compare your own with the rest of the team was a highly rated feature that was not yet implemented into the application.
- Join-code placement: many players expected the join code to appear on the first signup screen. The flow needed clearer signposting.
- Preference semantics: players were uncertain whether primary or secondary preferred positions carried different weight.
- Out-of-position indicators were misinterpreted by managers as positive highlights rather than warnings.
- A specific bug was detected where formation options hid the lineup name input, blocking further progress.

Takeaways and influence

Changes that we later implemented after the session included placing join-code entry earlier in the signup flow, clarifying preferred position semantics, redesigning the out-of-position indicator to avoid misreading, and fixing the layout bug that hid required inputs.

Online interaction - Coaching forum
Date: 4 November
Participants: 1 (Academy coach)

As part of our wider qualitative research, we posted about Coach Canvas in online coaching communities and received a detailed reply from an experienced, part-time academy coach in Spain who works with academy age groups (identity verifiable). The response offered practical insight into how an organised academy currently uses software, and it suggested which features are genuinely useful in a busy coaching workflow.

Response summary

I am a part-time coach at a good academy in Spain in U16 and U19 categories.
Our main need in team management is to keep a record as faithful as possible of minutes, ownership and call-ups of the players. This helps to argue with parents and to keep detailed monitoring of promising players.
The module to design tasks and sessions is not used because it is inconvenient on mobile and time consuming on the web. Spending an afternoon building a library of exercises could help, but I cannot convince anyone to do that.
The agenda and scheduling with notifications and attendance confirmations is very useful for categories where football is secondary and absenteeism is high.
We do not use tablets during sessions beyond specific video sessions.
For lineups we use LineUp11 and then share screenshots in coaches groups. I do not think we should pay for another app when the club already provides a management app.

Key takeaways

- Recording minutes, selection and call-ups is an important administrative need for academies, particularly where parental oversight and club partnerships exist. This suggests Coach Canvas should either offer simple, auditable attendance/minutes logging or provide a lightweight exportable report that managers can use in club meetings.

- Coaches will not spend valuable time authoring complex drills on mobile. A practical solution is to support a small, coach-editable library of reusable drills that can be composed in a single session on desktop or prefilled by the manager and then reused on mobile. Building import and template tools will increase uptake.

- The respondent values an agenda with notifications and the ability for players to confirm attendance. This reinforces our prioritisation of Schedule and Availability features as core flows.

- Coaches rarely use tablets during regular sessions. The UI should be phone-optimised, with larger tap targets and minimal need for tablet interfaces during live coaching.

- Clubs may already supply administrative apps and will be reluctant to pay for overlapping services. Coach Canvas must therefore demonstrate complementary value, offer frictionless sharing existing club systems, and consider a pricing model that respects club budgets.

Overall, the forum interaction reinforced our decision to concentrate development effort on our confirmed features, to offer small but well integrated tactical sharing functions, and to design for quick reuse rather than in-session composition of training materials.

Session 6: Final functionality test | Willows U16 coach
Date: 5 November
Participants: 1 (U16 coach)

Purpose and tasks
This late-stage test reviewed a near-complete prototype with features for club join, events, lineup creation, availability toggles and sharing.

Questions prepared:
- Could you try to create an Event for your next Match?
- How would you create a Lineup for this Match?
- Could you try swapping two Players?
- How easy did you find the process?
- What did you find the most tedious or uninspiring?
- Do you place value in being able to change the colours of the Club during setup?

Key tasks
- Create an event.
- Create and save a lineup.
- Swap players and examine bench and substitution flows.

Findings
The coach described the prototype as "really smooth" but found some domain knowledge gaps (for example, confusion about specific position abbreviations). Other observations included:

- The possibility to track match events in real time.
- Lineup “view mode” buttons that appeared disabled caused repeated taps and confusion.
- Bench and substitution behaviors needed clearer affordances and step-by-step microcopy.
- Colour customization was not a priority to this user.

Takeaways and influence
The biggest decision we took after the final testing session was the creation of a “Game Mode” where each Match event could have its own section, triggered by the Manager, where real-time statistics and Match Events can be recorded (including goals, assists, yellow/red cards and substitutions). This section added a layer of further depth to the base Coach Canvas application that we had developed until that moment.
UX and UI improvements were added in other sections, such as the complete removal of the “view mode” buttons in the Lineup creator, as they had been a product of concept prototyping, and the feature wouldn’t end up in the final artifact. Microcopy for substitution workflows was also improved, and a cosmetic color customization was de-prioritised in favour of a functional palette.

5.4. Cross-Session Analysis & Emergent Themes

Across six test events with 17 participants, core tasks were completed successfully in most cases with minimal facilitation. The naturalistic test conditions meant that full timing measurements were not recorded in every session, as it was unfeasible. Nevertheless, qualitative evidence indicates the following patterns:

- Core utility validated: participants consistently recognized the value of a single hub for schedule, squad and match lineups.
- Interaction expectation conflict: Despite drag-and-drop being a popular mental model pattern for the player swapping interaction, tap-to-swap was accepted after brief orientation, also being considered a simpler and more accessible alternative.
- Availability importance: in at least four sessions availability features were highlighted as essential managerial tools.
- Sharing requirement: in several tests coaches insisted on quick and on-the-go lineup sharing, making this a high priority feature.

5.5. Design Decisions Driven By Testing

1. ***Tap-to-swap lineup card interaction***
   Rationale: Tap-to-swap is reliable on low-end devices and with limited screen affordance. Optional dragging would be attractive but ultimately unnecessary.

2. ***Prominent availability indicators and attendance summarising iconography***
   Rationale: Managers requested quick-glance visibility of who is available. The Lineup view now includes color-coded dots and the Event view from the Home screen provides a compact availability summary such as "12 available".

3. ***Saved lineups and generation helper***
   Rationale: Saved lineups reduce repetitive work. Generation from availability helps managers produce a suggested starting eleven quickly by matching available players to their preferred positions.

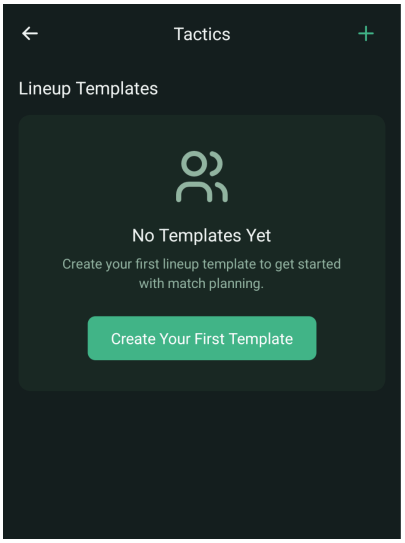

Figure 24: Lineup templates screen

4. ***Join-code visibility during signup and a quick-join option on Home***
   Rationale: Players expected to be able to enter join codes as soon as they start. The signup flow was adjusted to include an optional "Join a club" prompt early.

5. ***Lock-before-share and share-confirmation toasts***
   Rationale: Preventing accidental shares was a direct request. A lock toggle and explicit "share" confirmation toast were added.

6. ***Bench visibility and substitution clarity***
   Rationale: Coaches wanted substitutes always visible. The bench was expanded and substitution swapping flows were clarified with step hints.

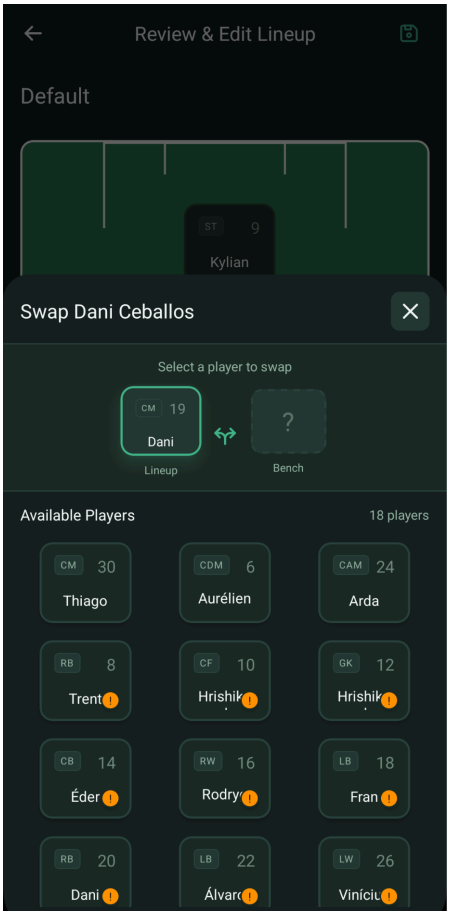

Figure 25: Bench and substitution logic

7. ***Microcopy, first-time hints and contextual tooltips***
   Rationale: Several users surprised by domain-specific terms needed just-in-time help. Short hints and an initial demo overlay were added to reduce friction.

8. ***Privacy defaults and minimal sensitive data collection***
   Rationale: Given youth teams, the app defaults to private player profiles (there is no way to view them unless you're inside the club) and minimises collection of personal details.

5.6. Technical & Methodological Reflections

Limitations of the testing programme

While the testing programme was effective in informing immediate design choices, several limitations must be acknowledged:

- Sample size and diversity: 17 participants is small and focused on local clubs and university players. Broader demographic sampling would strengthen external validity.
- Quantitative measurement: the project did not record consistent time-on-task or formalised SUS/NET PROMOTER SCORE metrics across sessions. Future work should incorporate standardised usability instruments for comparative evaluation.
- Longitudinal adoption: tests were single-session snapshots. Long-term adoption, retention and whether teams actually stop using WhatsApp require a longitudinal pilot.

Operational constraints

- Device variability: tests highlighted the need to account for older Android devices. Ensuring acceptable performance across a wide range of hardware will remain a development priority.
- Real-world interruptions: testing in training breaks meant sessions were brief and occasionally derailed. This context, while realistic, reduced opportunities for deep walkthroughs.

Recommendations for future testing

To address the limitations above, the next testing steps should include:

- Larger scale pilot with 6-10 clubs over a season to measure real-world adoption and time savings.
- Standardised SUS and task-timing measurements across a representative sample.
- A/B tests comparing tap-only, drag-only and hybrid defaults for lineup editing to quantify performance differences.
- Longitudinal interviews to understand retention and adoption friction points.

5.7. Conclusion

The iterative testing and analysis program for Coach Canvas demonstrates the value of short, contextual test cycles when designing for pragmatic users such as grassroots football coaches. Testing validated the central hypothesis: that a compact hub focused on Schedule, Squad and Upcoming Match lineups would deliver real value by reducing communication noise and manual coordination tasks. The programme also identified a set of consistent interaction expectations and constraints. Most importantly, tests translated directly into measurable product changes: defaulting to tap-to-swap with

optional drag, adding availability indicators and attendance summaries, implementing saved lineups and a generate-from-availability helper, improving onboarding for join codes, introducing lock-before-share and PNG export, and polishing bench and substitution workflows.

The test programme was pragmatic and user-centred, but it is only a first step. To claim robust external validity and to assess long-term impact on team performance and administrative effort, the app needs a wider pilot and standardised usability metrics. Nonetheless, the evidence gathered during the summer and autumn development cycles has produced a working prototype that addresses the principal pain points identified by managers and players and that can support further refinement in a staged deployment.

6. EVALUATION

6.1. Introduction

This section evaluates Coach Canvas against the project goals, reflecting on what worked well, what could have been improved, and where to focus next. It summarises critical success factors that determined the project's direction, defines the MVP we shipped, offers a multi-faceted application critique, and reviews teamwork, production and early marketing thinking.

6.2. Critical Success Factors

Success for Coach Canvas was measured by a set of practical, user-centred criteria that emerged from the research phase and guided development decisions:

- ***Usability and task efficiency***
  Can a manager create, edit and share a lineup within a short window of time (for example under five minutes)? Does the Schedule view allow the user to find the next event in one or two taps? These operational benchmarks were prioritised and validated through testing.

- ***Robustness across devices and connectivity***
  The app must be reliable on lower-end Android handsets and tolerate intermittent network connectivity. Offline-first behaviours and careful animation choices supported this requirement.

- ***Adoption potential***
  Early indicators such as participant enthusiasm, expressed willingness to replace WhatsApp for lineup sharing, and requests for player statistic tracking showed a clear real-world adoption potential.

- ***Live match-event tracking***
  Last-effort adoption of the "Game Mode" feature provided the most visible demonstration that the app could support a full match lifecycle and not just pre-game planning. It served both as a technical proof point and as a feature with tangible perceived value.

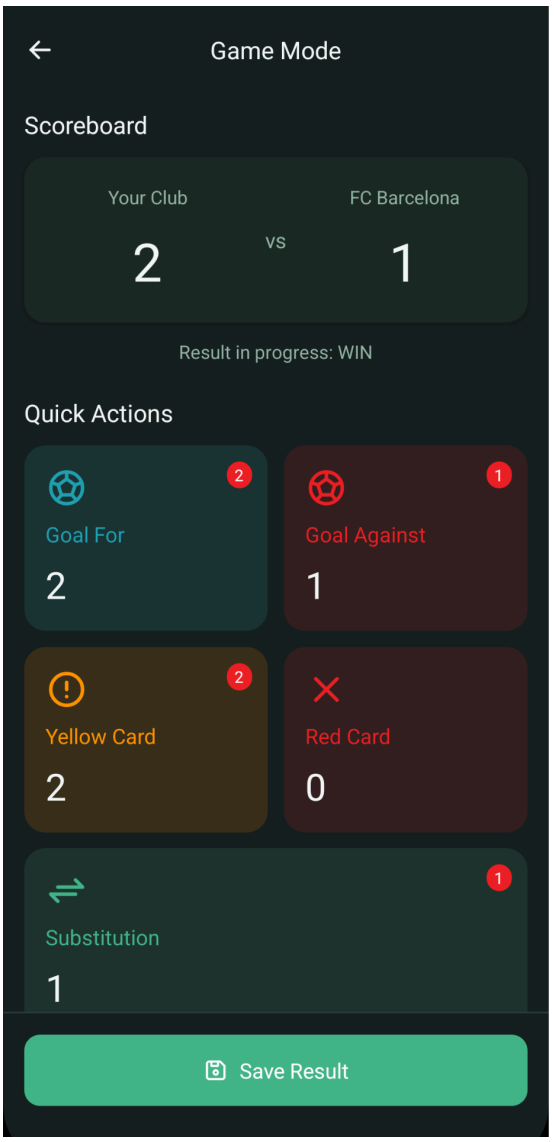

Figure 26: Game Mode or live match statistic tracking

6.3. Minimum Viable Product (MVP) delivered

For the final submission, the MVP focused on the highest value, most frequently used flows. Core capabilities included:

- Account creation and Google sign-in.
- Club creation and join via auto-generated code.
- Home hub with clear access to Schedule, Squad, Tactics, Upcoming Match and Match History.
- Schedule management with event creation and calendar/list views.
- Squad management with player cards, availability toggles, positional data and player statistics.
- A tactical canvas with lineup creation, saved lineups and a generation helper.
- Basic Game Mode for live match event tracking.
- Offline sync and Firestore-backed persistence for core data.

6.4. Design Critique

The visual direction deliberately prioritised legibility, accessibility and domain familiarity over bold art direction risks. The dark, green-anchored palette and Inter typeface work well for pitch-related semantics and sideline visibility conditions. Rounded shapes and vector icons promote clarity and low cognitive load.

For future releases, the design system could be extended with an optional skin or theme layer that introduces deeper customization while keeping the accessible baseline unchanged. Some elements had to be sacrificed in favour of usability or features, such as the pitch design within the lineup viewer, which was effectively left quite simplified as per risking functionality had it been edited further.

6.5. Teamwork & Production

Team collaboration was a clear strength. Our partnership combined complementary skills (front-end design and interaction leadership on the one hand, back-end engineering and stability focus on the other) and used a consistent cadence of weekly goal planning during summer and near-daily sessions during the project's final phase. Practical tools such as Trello, shared Figma files and a central GitHub repository kept our work coordinated. The main production constraint was access to test participants, given that tester recruitment was somewhat opportunistic and this limited the statistical robustness of some findings. Regardless of any constraints, we believe that the team delivered a cohesive product and maintained a pragmatic prioritisation of features.

6.6. Marketing & Social Media

No formal marketing was executed during development because product quality and stability were prioritised over "hype". However, we are aware that moving from prototype to pilot requires a focused outreach plan. Future marketing priorities should include compiling pilot testing offers to more local clubs, building social presence (Instagram, Facebook, LinkedIn, Twitter/X), promoting our incoming demo video, and creating a basic landing page that captures interest and signs up interested teams.
We believe brand values should emphasise time-savings, simplicity and the one-tap tactical sharing, which proved to be the strongest motivational driver in user tests.

6.7. Future Work

Interaction richness

Add a robust drag-and-drop mode in the lineup editor, and implement simple vector drawing for tactical arrows and transitions as optional advanced tactical tools.

Analytics and coaching aids

Introduce lightweight statistics dashboards (attendance trends, goal involvement, minutes played) and a simple tagging system so managers can quickly sort by last-attended training or position suitability.

Drill library and content

A curated, searchable drill database with coach-contributed content could increase the app's weekly utility and retention. Integrating short video or illustrations would add pedagogic value for managers and players.

Multi-manager workflows and governance

Formalise multi-manager roles within a club, invite tokens and establish a managerless recovery flow. These policies were initially prototyped but discarded, as they warrant further real-world validation and higher database complexity.

Localization and multi-sport adaptation

Consider localisation for key regions and explore adapting the core pattern to other field sports such as Gaelic games or rugby. Changing sports would require modest changes to pitch geometry and position semantics but could open valuable niche markets within Ireland.

6.8. Conclusion

Coach Canvas met its core ambitions by delivering a compact, usable system for lineup creation, scheduling and basic match tracking, with design decisions prioritising accessibility and reliability at the cost of some stylistic risk. The immediate next steps should be pragmatic; iterate on interaction affordances with controlled A/B tests, and prepare a communication and marketing plan that leverages the app's strongest value propositions. With careful prioritisation and measured growth, we believe that Coach Canvas has a clear path as a sustainable tool for grassroots sport.

7. CONCLUSION

This project set out to create Coach Canvas, a lightweight mobile application that consolidates the essential tasks of grassroots team management. Across the development cycle we balanced the two competing priorities of an app that had to be simple and reliable enough for regular use by busy amateur managers and part-time players, and also provide meaningful tactical affordances that replace traditional whiteboards and fragmented messaging workflows. By delivering a working prototype that supports club creation, event scheduling, availability reporting, lineup creation and basic match tracking, we met the original aims in a way that is demonstrably useful to the intended user base.

Three critical success factors guided design and implementation:

- ***Usability:*** the app must be learnable and fast to use in time-pressured contexts. This informed choices such as a compact Home hub, large tap targets, explicit availability indicators, and the tap-to-swap interaction as the default for lineups.
- ***Reliability:*** real-world amateur clubs operate under device and connectivity constraints, so the architecture favoured resilient patterns, typed service modules and Firestore transactions for sensitive operations like club creation and squad-number reservations.
- ***Iterative validation:*** multiple in-context test sessions with coaches and players repeatedly shaped priorities, exposing interaction expectations and pragmatic workarounds that were folded into subsequent development. Those tests also confirmed the product-market fit of the core concept, namely a single hub for schedule, squad and match lineups.

The MVP delivered focuses tightly on the manager-player workflow. Managers can create clubs, generate join codes, create events and craft match-specific lineups. Players can join clubs, set availability and view manager-created lineups, as well as their own and other players' statistics. This scope intentionally omits heavyweight analytics, which would add complexity and risk for the core user group but ensures that the features included are robust, accessible and aligned with user needs identified during field testing.

Testing played a central role in shaping the final product, with sessions used on training breaks and beside pitches to capture real behaviour, that uncovered several high-value insights, such as the interaction expectations, the strong demand for quick sharing, and the necessity of clear onboarding for join code flows. Each observation led to concrete changes, and various UI microcopy improvements were implemented. The testing process also surfaced project constraints, such as the difficulty of recruiting sustained testing time from clubs during busy seasons, which limited the breadth of quantitative timing measures.

From a technical perspective, the codebase followed a disciplined approach, making the UI layer simpler to reason about and more resilient to backend changes, as well as ensuring performance by improving perceived speed on lower-end devices. Security

rules and role-aware capabilities were implemented so that Managers and Players have narrowly scoped permissions, aligning technical controls with the product mental model.

Sadly, some interaction richness was sacrificed for reliability. Drag-and-drop lineup editing, freehand tactical drawing and a full drill tutorial environment were deferred to future work, because they either required extensive device-specific testing or, for the most part, introduced disproportionate technical risk. Similarly, advanced multi-manager governance flows and "manager-less" recovery patterns were prototyped conceptually but moved out of the MVP because they substantially increase data and policy complexity.

Looking forward, there is a clear roadmap to build on the foundation delivered. Short term priorities include a careful interaction testing programme, incremental improvement of bench and substitution affordances, and a lightweight analytics dashboard for specific competitions created by the player. Medium term priorities would be a coach-editable drill library to reduce the friction of session planning, richer export options for clubs that already use administrative suites, and a polished onboarding funnel with targeted marketing collateral for local pilot clubs. Longer term, Coach Canvas could explore multi-sport variants and partnerships with local leagues, provided the product retains its core promise of being simple and fast.

In summary, Coach Canvas demonstrates how a focused, user-centred approach can produce practical tools for grassroots sport. The project shows that modest technological investment in well-chosen flows yields real benefits for time-poor coaches and semi-committed players. The discipline of iterative testing and modular engineering has left the project in a position where careful extension and measured piloting can meaningfully expand its scope without sacrificing the usability that makes it valuable to its users.

BIBLIOGRAPHY & REFERENCES

Andrew Heim. (2025). *React Native Expo Firebase Tutorial: Your First App* [Video]. YouTube. https://www.youtube.com/watch?v=a0KJ7l5sNGw

Code with Beto. (2023, June 20). *Uploading videos & images to Firebase Storage* (project page). https://codewithbeto.dev/projects/firebase-storage

freeCodeCamp.org. (2021, October 13). *Firebase Cloud Firestore – Database Crash Course*. freeCodeCamp. https://www.freecodecamp.org/news/firebase-firestore-crash-course/

Iconify. (2025). *Iconify — home of open source icons*. https://iconify.design/

Net Ninja. (n.d.). *Firestore tutorial for React Native* [Video]. YouTube. https://www.youtube.com/watch?v=Al5qBOhSfjE

PedroTech. (n.d.). *Firebase React Course for Beginners – Learn Firebase V9+ in 2 Hours* [Video]. YouTube. https://www.youtube.com/watch?v=2hR-uWjBAgw

Pixabay. (n.d.). *Pixabay – Free images & royalty-free stock*. https://pixabay.com/

React Navigation. (n.d.). *Getting started* (documentation). https://reactnavigation.org/docs/getting-started/

React Native. (2025). *Getting started* (documentation). https://reactnative.dev/docs/getting-started

React Native community / Stack Overflow (aggregated tags). (n.d.). *React Native* (Stack Overflow tag page). https://stackoverflow.com/questions/tagged/react-native

React Native community / Stack Overflow (aggregated tags). (n.d.). *Firebase* (Stack Overflow tag page). https://stackoverflow.com/questions/tagged/firebase

React Navigation community / Stack Overflow (aggregated tags). (n.d.). *React Navigation* (Stack Overflow tag page). https://stackoverflow.com/questions/tagged/react-navigation

fsniraj. (n.d.). *React Native role-based authentication* [Video]. YouTube. https://www.youtube.com/watch?v=F7qo3tcSWP8

Coolors. (n.d.). *Coolors – color palette generator*. https://coolors.co/

Expo Documentation. (n.d.). *Expo docs*. https://docs.expo.dev/

Firebase. (n.d.). *Firebase documentation*. https://firebase.google.com/docs

Firebase. (n.d.). *Cloud Firestore documentation*. https://firebase.google.com/docs/firestore

PedroTech (channel playlist). (n.d.). *Firebase V9+ / React Firebase playlist* [Playlist]. YouTube. https://www.youtube.com/watch?v=2hR-uWjBAgw

"React Native Image Picker & Firebase Storage" (Code with Beto / YouTube). (n.d.). *React Native image upload tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=cq5TGA6sBQQ

Stack Overflow. (n.d.). *Role-based authentication in React Native* (Q&A). https://stackoverflow.com/questions/67178018/role-based-authentication-in-react-native-app-using-firebase

Reddit. (n.d.). *r/reactnative* (community). https://www.reddit.com/r/reactnative/

Reddit. (n.d.). *r/Firebase* (community). https://www.reddit.com/r/Firebase/

Reddit. (n.d.). *r/MobileDev* (community). https://www.reddit.com/r/MobileDev/

Iconify (icon docs). (2025). *Iconify – Getting started and icons documentation*. https://icon-sets.iconify.design/ and https://iconify.design/docs/

Google Fonts. (n.d.). *Inter* (typeface). https://fonts.google.com/specimen/Inter

Pluralsight (guide). (2020, October 17). *Integrate Firebase Storage and Image Picker in React Native* (guide). https://www.pluralsight.com/resources/blog/guides/integrate-firebase-storage-and-image-picker-in-react-native

React Navigation. (n.d.). *Configuring links / deep linking* (documentation). https://reactnavigation.org/docs/configuring-links/

Expo Linking. (2025). *Linking into other apps* (documentation). https://docs.expo.dev/linking/into-other-apps/

Google Cloud (Firestore REST reference). (2025). *Cloud Firestore API reference*. https://docs.cloud.google.com/firestore/docs/reference/rest