DIGI4000A

CREATIVE PROJECT

PROJECT DETAILS

APP: ELEVYN

BY: OFENTSE MAGABEANE

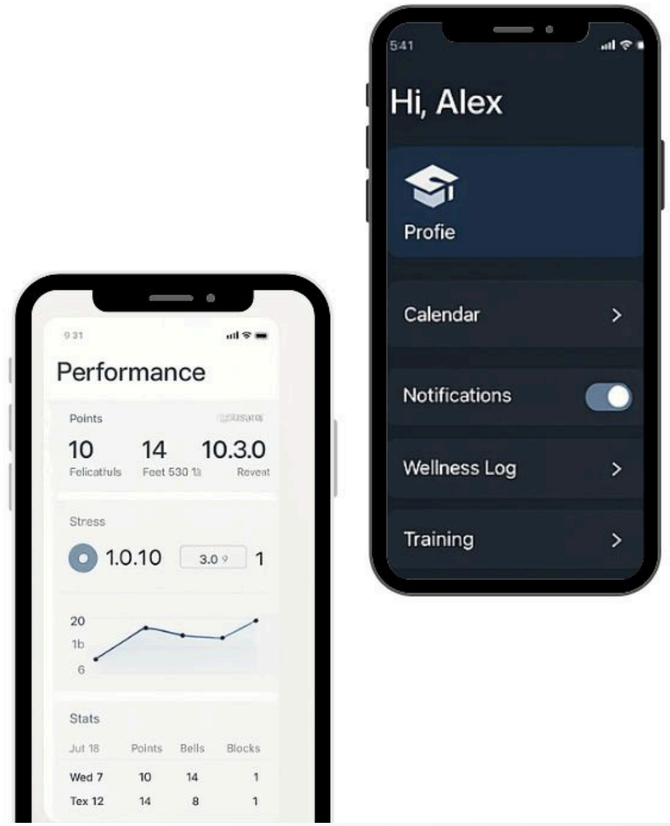
PRESENTATION
CONTENTS:

APP NAME:
ELEVYN

Elevyn is a mobile-first app designed to support student-athletes in managing both their academic and athletic responsibilities within a single, streamlined platform. The app bridges the gap between existing learning management systems (LMS) and sports tracking tools by offering an integrated, user-friendly experience that reflects the dual life of a student-athlete.

Elevyn

Student-athlete lifestyle app



Why is it important to create this :

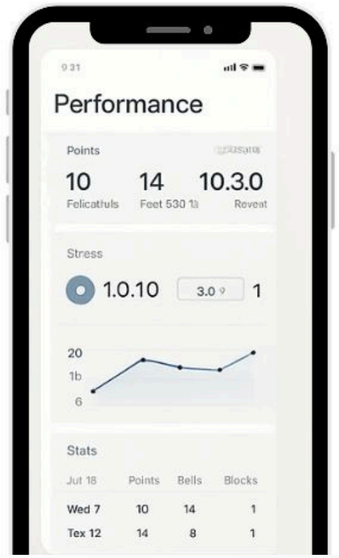
- Student-athletes are overwhelmed juggling performance, school, and recovery.
- No major app balances both academic and athletic needs in a unified platform.
- Helps athletes build discipline and balance — not just performance.
- Ideal for personal development, sports organisations, and school athletic programs.

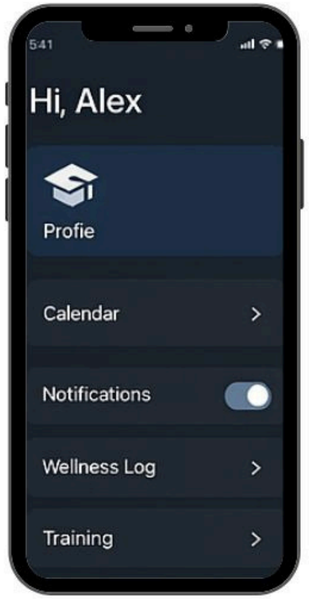
What the App Represents:

- Student–Athlete dual life (balance, time management)
- Performance + wellness + academics in one space
- Built for growth, clarity, and control
- Designed with empathy and intention

Elevyn

Student-athlete lifestyle app

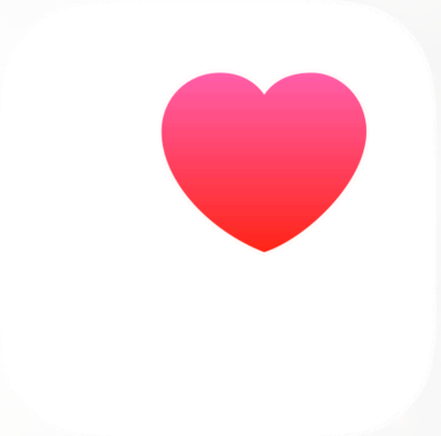




Sources of Inspiration

Headspace

StatsAssistant

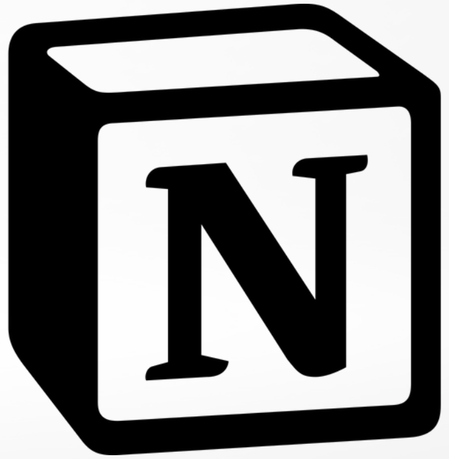


Health

Nike Training Club

Nike Run Club

uLwazi (Canvas)



Notion

COMPETITOR RESEARCH

In an increasingly saturated app ecosystem, student-athletes remain underserved. While there are powerful tools for fitness, wellness, or academics, few — if any — speak to the interconnected lifestyle of student-athletes who must simultaneously manage academic deadlines, athletic performance, and personal wellbeing. This is where Elevyn emerges as a unique solution: a holistic, mobile-first companion built specifically for student-athletes.

Nike Run Club / Nike Training Club

These apps offer excellent guidance on physical fitness, personalised workouts, and running metrics. However, they focus solely on athletic performance, without addressing academic obligations or mental health. They cater to general users and professional athletes — not the dual responsibilities of students balancing training with coursework.

Headspace

Headspace provides a strong foundation for mindfulness, meditation, and mental health. It is well-designed, easy to use, and supportive of emotional wellbeing. Yet, it does not offer integration with academic or athletic tracking, and lacks contextual relevance to the student-athlete experience.

Notion / Canvas LMS

Both Notion and Canvas are widely used in academic settings. Notion excels in task management and flexibility; Canvas supports learning management and course tracking. Still, these tools lack intuitive support for athletic planning, and do not incorporate any features related to physical performance or wellness.

COMPETITOR RESEARCH

Basketball Stats Assistant

These tools specialise in game-day statistics, team management, and season performance analysis. While useful for coaches and athletes, they tend to focus narrowly on in-game metrics, and exclude off-court lifestyle factors like sleep, nutrition, or academic scheduling.

Health Apps (e.g., Apple Health, Samsung Health)

Generic health apps track daily activities like sleep, water intake, and heart rate. However, they operate in silos, do not personalise data for the student-athlete context, and offer little to no planning or academic support. The experience is often fragmented and passive.

Elevyn offers an integrated, seamless experience where student-athletes can manage their dual- life commitments. Unlike other apps, Elevyn does not require the user to jump between platforms. Everything is in one place — balanced, intuitive, and tailored for the demands of dual life as a student and athlete. It supports both preloaded mock data and lightweight user input to demonstrate functionality during early phases, with a clear roadmap toward scalable features.

ELEVYN
FEATURES

Mock Authentication (Intro Screen)

- What: Simple login screen with a hardcoded/dynamic user ("Jamie").
- Purpose: Simulates user-specific experience.

Dashboard (Home Screen)

Summary tiles with a blend of academic & athletic info:

- Upcoming event: "Assignment due tomorrow"
- Next game: "Vs Wits on Thursday"
- Training: "3 sessions this week"
- Academic task: “You have 2 tasks outstanding”
- Wellness highlight: “Your energy was low on game day”

Integrated Calendar View

Academic + Athletic Events Combined:

- 📚 "Research Methods Assignment – Due Tuesday"
- 🏀 "Basketball Game – Thursday"
- 🛌 "Recovery Day – Sunday"

Preloaded data only and dynamically added dats (a mixture of preloaded data and data that the user adds)

Weekly Planner / Task Manager (Academic + Sport)

- User can edit schedule
- Tasks have categories/tags: 🏀 Athletic or 📚 Academic
- Tasks show status: “To do” or “Done”

Examples:

- 📚 “Submit Lit Review”
- 🏀 “Watch game film”
- 📚 “Meet tutor at 3pm”

ELEVYN
FEATURES

Wellness Log (Academic + Athletic Impacts)

- Track energy, stress, sleep — can relate to school, sport, or both
- ✅ User inputs daily data
- Visual insights: “Stress levels highest during game + deadline week”

Progress Dashboard

A holistic view:

- Academic Tasks Done: 5/7 this week
- Athletic Events Attended: 3 trainings, 1 game
- Wellness Score: 72% energy average
- Balance Insight: “You’ve logged more hours on sport than study this week — try adjusting.”

Settings/Profile

- Just visual — change avatar, sport, major, Light/dark mode toggle
- Potential future settings: notification preferences, reminders

Performance Tracker (Athletic)

- Preloaded game stats (3 basketball games)
- No user input for now
- Graphs + summary (FG%, points over time)

FEATURE **UPDATES:**

Dashboard:

- Progress Dashboard will be merged with the main dashboard so that the feature is more useful and doesn't feel redundant.
- Instead of having two dashboards- having one main works well and makes the demo feel less overwhelming

Settings:

- A light/dark mode toggle feature included to offer more customisation to the app demo.
- Potential future setting have been included in the app demo iterations: a mock toggle feature has been used to demonstrate how it would work in future.

Wellness Log:

- Wellness log includes a "wellness resources" section which includes: guided meditations, Nutrition tips.
- Users can input their data and save the log each day.

PROJECT **TIMELINE:**

1

Phase 1: Concept

- Finalize branding, logo, colours
- Create wireframes and user flow
- Build basic screens in Expo (login, dashboard, 1-2 key features)
- Load sample data

2

Phase 2: Feature Development

- Finish all key screens (calendar, planner, performance, wellness, dashboard)
- Add more interactivity (task completion, wellness check-ins)
- Use local data for persistence
- Polish navigation + animations

3

Phase 3: Polish + Backend

- Refine UI and transitions
- Optional: Add Firebase backend
- Conduct user testing (even if just 2–3 people)
- Collect feedback and improve UX

4

Phase 4: Final Touches

- Final bug fixes and testing
- Prepare documentation, README, and project summary
- Record complete demo walkthrough video
- Pitch Deck : for future development

PROJECT
HOURS:

Task Tracker: Planning, Research, & Wireframes

Task	Status	Hours
Competitor research & feature scoping	✅ Complete	8
User Journey Mapping	✅ Complete	6
UI Mood-boarding & Design system	✅ Complete	6
Initial Sketches & low-fidelity wireframes	✅ Complete	6
Mid-High Fidelity Wireframes	✅ Complete	10
Finalising features & roadmap	✅ Complete	4

PROJECT
HOURS:

Task Tracker: Front-End (React Native & Expo)

Task	Status	Hours
Project setup (Expo, Git, folder structure)	✅ Complete	8
Navigation (React Navigation setup)	✅ Complete	10
Authentication flow (mock login/signup)	👀 Under review	10
Dashboard UI (calendar, performance tiles)	👀 Under review	20
Calendar feature (preloaded + editable)	✅ Complete	10
Performance tracking UI (mock stat display)	✅ Complete	18
Wellness log UI (mood, sleep, input form)	✏️ In progress	16
Settings page + mock profile	✅ Complete	8
Reusable components & styling cleanup	👀 Under review	8

PROJECT
HOURS:

Task Tracker: Feature Integration + Data Storage

Task	Status	Hours
Connect Front-end to Back-end	✏️ In progress	10
Integrated calendar data into UI (Front and Back end)	👀 Under review	14
Populate preloaded data	✅ Complete	6
Planner/Calendar data (Add, Delete, Edit)	✅ Complete	8
Wellness Log Features (Inputs and Charts)	✅ Complete	8
Settings toggle (Light/Dark Mode toggle, Profile etc.)	✅ Complete	4

PROJECT
HOURS:

Task Tracker: Back-End

Task	Status	Hours
Backend setup + environment config	✅ Complete	6
Design mock database schema	👀 Under review	10
Build mock user authentication API	✏️ In progress	10
Endpoints for calendar, planner, stats	⏳ Not started	10
Wellness log & academic data APIs	✏️ In progress	14
Local data storage & retrieval logic	✏️ In progress	12
API testing	⏳ Not started	8
Basic error handling + response formatting	⏳ Not started	6

PROJECT
HOURS:

Task Tracker: UI Polish + Accessibility

Task	Status	Hours
Design System Cleanup (colours, fonts, spacing)	✏️ In progress	8
Add Icons, visuals, empty states	👀 Under review	6
Responsive design	👀 Under review	6
Accessibility (alt text, navigation)	⏳ Not started	8
UI Animations	⏳ Not started	6

PROJECT
HOURS:

Task Tracker: Testing + Bug Fixing

Task	Status	Hours
Manual testing (Feature by Feature)	🔄 Constantly testing	12
Cross- device & OS testing	🔄 Constantly testing	10
Fixing common UI & Logic bugs	👀 Under review	6
Code cleanup & commenting	⏳ Not started	8

PROJECT HOURS:

Task Tracker: Documentation + Demo Prep

Task	Status	Hours
Write README (Feature, install, usage)	Not started	6
Record walkthrough (Demo)	Constantly updating	8
Prepare Slides for project presentation	Constantly updating	8
Write Developer notes for future dev	Not started	10
Backup and project folder organisation	Not started	6
Polish Demo	Not started	8

DEVELOPER **NOTES:**

Code review: The Dev notes presented here are notes of code reviews (what the code is doing and where it could improve) with screenshots of the code. The code review will also include explanation of the screens chosen for the review- essentially explaining what the feature is supposed to do.

FRONT-END

- Progress Dashboard (will be converted to main dashboard)
- Calendar
- Navigation

BACK-END

- User Authentication (This is the only back-end concept that works without throwing errors)

DEVELOPER NOTES:

PROGRESS DASHBOARD

progressdashboard.tsx

app > screens > progressdashboard.tsx > ProgressDashboard

```tsx
const ProgressDashboard = () => {

        {/* Academic Performance */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Academic Tasks</Text>
          <Text style={styles.valueText}>
            {academicTasks.done}/{academicTasks.total} completed this week
          </Text>
        </View>

        {/* Athletic Events */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Athletic Events</Text>
          <VictoryChart width={350} height={250}>
            <VictoryLine
              data={[
                { x: 1, y: 2 },
                { x: 2, y: 3 },
                { x: 3, y: 5 },
                { x: 4, y: 4 },
              ]}
              style={{
                data: { stroke: "#1D2D44", strokeWidth: 3 },
              }}
            />
          </VictoryChart>
          <Text style={styles.valueText}>
            {athleticEvents.trainings} trainings, {athleticEvents.games} games
          </Text>
        </View>

        {/* Balance Insight */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Balance Insight</Text>
          <Text style={styles.insightText}>{balanceMessage}</Text>
        </View>
      </ScrollView>
    </View>
  );
```

codedbyfefe (1 week ago) Ln 34, Col 47 Spaces: 2 UTF-8 LF TypeScript JSX Prettier

progressdashboard.tsx

app > screens > progressdashboard.tsx > ProgressDashboard

```tsx
const ProgressDashboard = () => {

        {/* Academic Performance */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Academic Tasks</Text>
          <Text style={styles.valueText}>
            {academicTasks.done}/{academicTasks.total} completed this week
          </Text>
        </View>

        {/* Athletic Events */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Athletic Events</Text>
          <VictoryChart width={350} height={250}>
            <VictoryLine
              data={[
                { x: 1, y: 2 },
                { x: 2, y: 3 },
                { x: 3, y: 5 },
                { x: 4, y: 4 },
              ]}
              style={{
                data: { stroke: "#1D2D44", strokeWidth: 3 },
              }}
            />
          </VictoryChart>
          <Text style={styles.valueText}>
            {athleticEvents.trainings} trainings, {athleticEvents.games} games
          </Text>
        </View>

        {/* Balance Insight */}
        <View style={styles.card}>
          <Text style={styles.cardTitle}>Balance Insight</Text>
          <Text style={styles.insightText}>{balanceMessage}</Text>
        </View>
      </ScrollView>
    </View>
  );
```

codedbyfefe (1 week ago) Ln 34, Col 47 Spaces: 2 UTF-8 LF TypeScript JSX Prettier

DEVELOPER NOTES:

PROGRESS DASHBOARD

Code review:

Separation of Styles:

- Styles are imported from a dedicated styles/progressdashboardstyles file, keeping UI and logic separate.

Logical Structure:

- The component is well-structured and broken into clearly defined sections (Wellness, Academics, Athletics, Insight).
- The current data is hardcoded. To make the component scalable or testable, consider passing props or fetching from the backend

Use of VictoryNative:

- Good use of data visualization through VictoryPie and VictoryLine.

Semantic Naming:

- Variables like academicTasks, athleticEvents, wellnessScore, and balanceMessage are well-named and self-explanatory.

Feature breakdown:

This feature was a standalone feature but will be merged into the Home dashboard to make the feature seem less redundant- the current home dashboard is plain and doesn't contain any useful information. This dashboard displays an genral overview of academics (tasks complete), the user's overall wellness score as well as athletic events and balance insights (how well they are balancing the commitments.

VictoryChart Line Graph Data:

- Problem: Data is hardcoded and might confuse the user if it doesn't match actual athleticEvents data.
- Suggestion: Use athleticEvents to create real or pseudo-randomized time-series data, or clearly label this as a "sample trend".

DEVELOPER NOTES:

CALENDAR SCREEN

calendar.tsx

app > screens > calendar.tsx > CalendarScreen > handleDeleteEvent > onPress

```tsx
export default function CalendarScreen() {
  const router = useRouter();

  // Events are stored as an object: { "YYYY-MM-DD": ["event1", "event2"] }
  const [events, setEvents] = useState<Record<string, string[]>>({
    "2025-09-14": ["Training: Morning gym session", "Study Group"],
    "2025-09-15": ["Lecture: Data Structures class"],
  });

  const [selectedDate, setSelectedDate] = useState("");
  const [modalVisible, setModalVisible] = useState(false);
  const [newEvent, setNewEvent] = useState("");
  const [editingIndex, setEditingIndex] = useState<number | null>(null);

  // Load events from storage
  useEffect(() => {
    const loadEvents = async () => {
      try {
        const storedEvents = await AsyncStorage.getItem("events");
        if (storedEvents) {
          setEvents(JSON.parse(storedEvents));
        }
      } catch (error) {
        console.error("Error loading events", error);
      }
    };
    loadEvents();
  }, []);

  // Save events to storage
  useEffect(() => {
    const saveEvents = async () => {
      try {
        await AsyncStorage.setItem("events", JSON.stringify(events));
      } catch (error) {
        console.error("Error saving events", error);
      }
    };
```

codedbyfefe (1 week ago) Ln 88, Col 56 Spaces: 2 UTF-8 LF TypeScript JSX Prettier

calendar.tsx

app > screens > calendar.tsx > CalendarScreen > useEffect() callback > saveEvents

```tsx
export default function CalendarScreen() {
  return (
    <View style={styles.container}>
      <Text style={styles.header}>📅 Training & Academic Planner</Text>

      {/* Calendar */}
      <Calendar
        onDayPress={(day) => setSelectedDate(day.dateString)}
        markedDates={{
          ...Object.keys(events).reduce((acc, date) => {
            acc[date] = { marked: true, dotColor: "#A6E1FA" };
            return acc;
          }, {} as Record<string, any>),
          [selectedDate]: {
            selected: true,
            selectedColor: "#1D2D44",
          },
        }}
        theme={{
          backgroundColor: "#111315",
          calendarBackground: "#111315",
          textSectionTitleColor: "#A7A9AB",
          dayTextColor: "#FFFFFF",
          monthTextColor: "#A6E1FA",
          selectedDayBackgroundColor: "#1D2D44",
          arrowColor: "#A6E1FA",
        }}
      />

      {/* Events List */}
      {selectedDate ? (
        <View style={styles.eventBox}>
          <Text style={styles.eventTitle}>Events on {selectedDate}:</Text>
          {events[selectedDate]?.length ? (
            events[selectedDate].map((event, idx) => (
              <View key={idx} style={styles.eventRow}>
                <Text style={styles.eventText}>{event}</Text>
                <TouchableOpacity
```

codedbyfefe (1 week ago) Ln 51, Col 8 Spaces: 2 UTF-8 LF TypeScript JSX Prettier

DEVELOPER NOTES:

PROGRESS DASHBOARD

Code review:

- The calendar supports adding, editing, and deleting events per date — core features are well implemented.
- Modal interaction is intuitive; calendar markings and themes are clear and visually coherent.

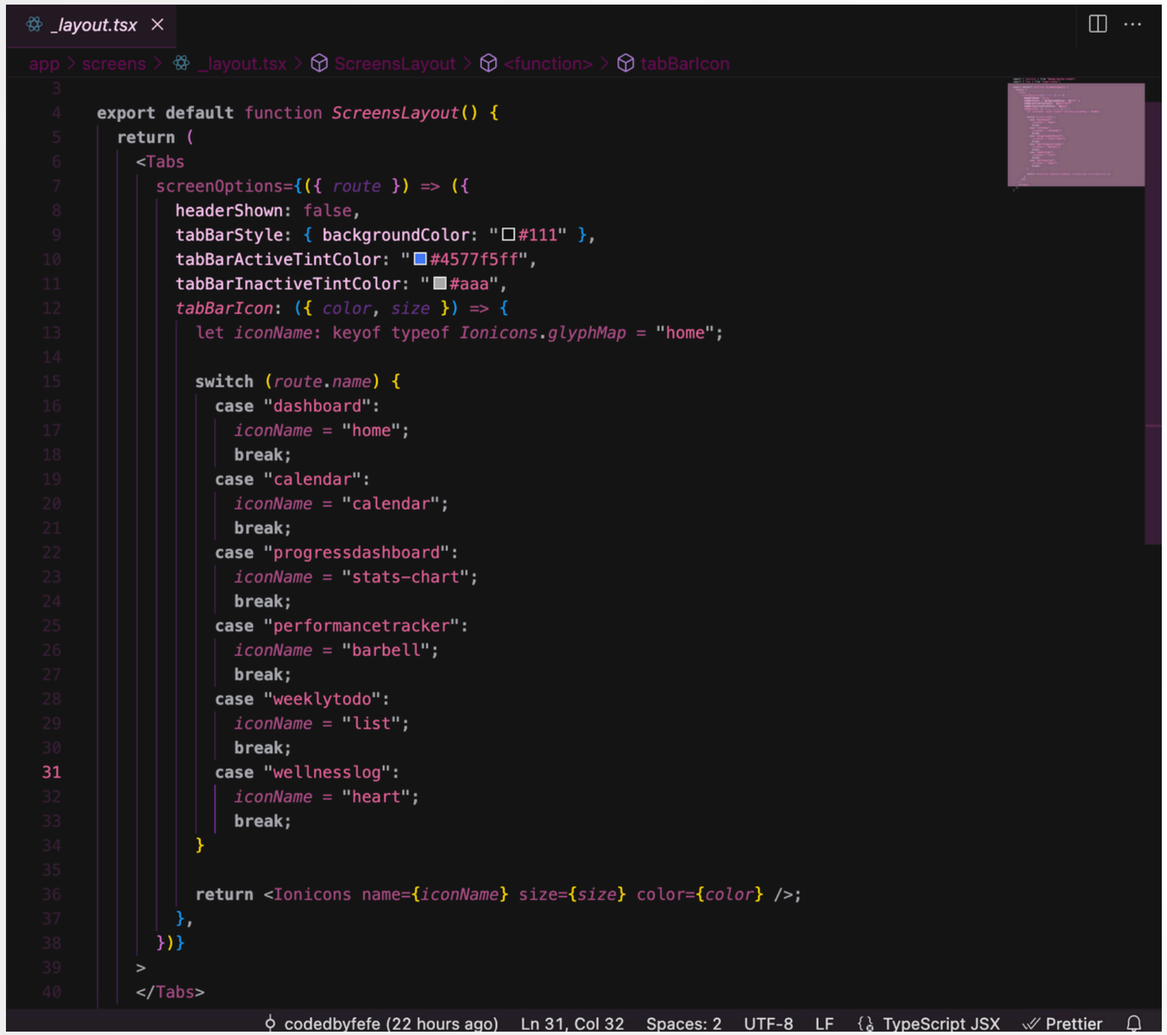
- Logical and easy to follow; good naming conventions (they are self explanitory)
- For scalability, wrap handlers like handleSaveEvent, handleDeleteEvent, etc., with useCallback. This prevents unnecessary re-renders if you extract event rows into child components later.
- The code readability needs to be reviewed adn evaluated

Feature breakdown:

The Calendar feature serves as a schedule manager/view: it displays the users schedule (class, training, due dates)- it will use preloaded data and dynamically added data(the user can add and delete events). Back-end implementation will be added to match any inputs and stored data from the user's inputs.

DEVELOPER NOTES:

NAVIGATION: BOTTOM TABS



Code review:

- `The switch statement works, but can be cleaner and more scalable with an object map: This approach is more maintainable, cleaner and avoids verbose switch syntax and easier to extend later.
- Accessibility need improvement
- The navigation is still hard coded and this might need to be reviewed if you want to use more of the back- end for the navigation as well.
- Review the Tabs component: it was noted that the component is empty (the code works but its not well written and needs to be improved).

PROJECT GOAL:

Personal goals:

- Improved confidence in full-stack mobile development.
- Better understanding of how to scope and manage a solo product build from scratch.
- Refined skills in UI/UX communication, visual hierarchy, and cross-disciplinary app thinking.

Professional goals:

- Improve UI/UX design thinking, including colour theory, typography, wireframing, and prototyping in Figma.
- Create a polished, real-world portfolio piece that demonstrates end-to-end product thinking: from ideation to deployment.
- Explore ways to ethically and mindfully represent underrepresented groups (e.g., African student-athletes) in app design and data models.

Project specific goals:

- Design and build a cross-platform mobile app using React Native for the front-end and Python (Flask or FastAPI) for the back-end.
- Provide integrated wellness tracking, including sleep, hydration, mood, and mental wellness.
- Incorporate academic planning tools, such as a calendar and task log, to support scheduling around training and game days.
- Offer visual performance insights using charts and logs to show patterns in both academic and physical performance.
- Ensure the app is accessible, user-friendly, and visually aligned with the needs of student-athletes (e.g., bold visuals, dark mode, intuitive navigation).
- Maintain data integrity and scalability with future API/database integration.

SUCCESS CRITERIA:

A successful outcome for this project will be defined by the following:

1. Functional Completion
   - All core screens (auth, dashboard, calendar, stat charts, input logs) are designed, prototyped, and implemented.
2. Design Coherence
   - A consistent visual language across dark/light modes, aligned with the brand identity (e.g., logo, typography, palette).
3. User-Centered Flow
   - Seamless onboarding and navigation with minimal cognitive load.
4. Extensibility
   - The app is built with modularity in mind, ready to support future features like AI feedback, performance predictions, or social accountability.

RUBRIC

Technical: Coding fundamentals (Clean code, performance, functionality, readability)	**25%**
Design: Responsive design, Control & Navigation, Information hierarchy)	**25%**
Research: Competitor research, development research (ie. design principles for mobile app design, user research)	**25%**
Aesthetics: Creativity, appealing visual design, coherent visual design)	**25%**

THANK YOU