SOFTWARE REQUIREMENT SPECIFICATION

Rome was Built in a Day

HIIT GYM MANAGER

GOOD X SOFTWARE

- Vansh Sood (u23534402)
- Denis Woolley (u23528860)
- Jared Hürlimann (u23543932)
- Jason Mayo (u23587572)
- Amadeus Fidos (u22526162)

Introduction

The HIIT Gym Management Software addresses a critical business need for HIIT-focused fitness studios: simplifying complex, error-prone administrative workflows so owners and coaches can concentrate on delivering high-quality training. Without specialized tools, gym operators juggle membership sign-ups, billing, class scheduling, attendance tracking, and member communications manually- tasks that consume time, introduce costly mistakes, and undermine member satisfaction.

This project will deliver a unified, user-friendly platform that centralizes every aspect of HIIT gym operations.

- **Members** will be able to register, manage their subscriptions, view and book classes, and track workout results from a mobile app.
- **Coaches** gain an intuitive interface for publishing daily workouts, overseeing assigned sessions, and reviewing performance data.
- **Managers** and administrators access a web-based dashboard to oversee pending and active members, configure weekly class schedules, assign coach roles, and monitor capacity and attendance trends.

By integrating automated payment processing, real-time leaderboards, and targeted notifications, the system not only streamlines daily tasks but also drives member engagement and retention. Leveraging data insights -such as attendance reports, performance trends, and class popularity- gym operators can make informed decisions to grow their business sustainably. In scope for this MVP are user authentication and role management, membership credit handling, class management and booking, workout submission, and basic administrative overviews. Subsequent iterations will expand to include advanced analytics, gamification features, and kiosk-style on-site displays.

By adhering to agile practices and CI/CD guidelines, our team will gain hands-on experience in modern web/mobile frameworks, PostgreSQL schema design, background job processing, and secure API development.

Deliverable:
A deployed, fully functional HIIT Gym Manager system comprising

1. A background worker daemon for scheduled tasks and notifications
2. A RESTful API backend
3. A React Native mobile app for members and coaches
4. A Next.js web app for managers and administrators

Functional Requirements Specification

Project Title: CrossFit Gym Management App
Client: Johan Bloem's CrossFit Box
Mentor: Johan Bloem
Delivery Deadline: 24 October 2025
Platforms: Mobile (Members & Coaches), Web (Managers/Admin)

R1: Authentication & Role-Based Access

R1.1: User Registration & Login

- R1.1.1: Members register and login via mobile (email, password, OTP).
- R1.1.2: Coaches login via mobile (provisioned by manager).
- R1.1.3: Managers & Admins login via web (single shared manager account + individual admin accounts).

R1.2: Logout & Session Management

- R1.2.1: All roles can logout.

R1.3: Role Management (Manager/Admin only)

- R1.3.1: Create, edit, or revoke Coach and Admin accounts.
- R1.3.2: Assign roles to users; enforce permissions per role.

R2: Member Onboarding & Profile

R2.1: Digital Waivers & Health Questionnaires

- R2.1.1: New members complete electronic waivers before booking.
- R2.1.2: Waiver signed, timestamped, stored securely.

R2.2: Member Profile

- R2.2.1: View/edit personal details, emergency contact.
- R2.2.2: Upload profile photo, track attendance history and PRs.
- R2.2.3: Choose to opt-in or opt-out of public leaderboards

R3: Class Booking (High Priority)

R3.1: Class Catalogue & Details

- R3.1.1: View upcoming classes (date, time, coach, capacity, workout type).
- R3.1.2: Search & filter by workout type, coach, time slot.

- R3.1.3: Join live classes when they start

R3.2: Member Booking & Cancellation

- R3.2.1: Book classes against active subscription or class-credit balance (enforce pay-first model).
- R3.2.2: Cancel within configurable window (e.g., ≥2 hrs before).
- R3.2.3: Prevent double-booking (same time slot) and capacity overflow.
- R3.2.4: In-app calendar integration & push/SMS reminders.

R4: Class Management & Setup (High Priority)

R4.1: Class Setup & Scheduling (Manager)

- R4.3.1: Create, edit, delete single or recurring classes (with presets) i.e. recurring weekly schedules or individually created classes
- R4.3.2: Override for holidays, coach swaps, special events.
- R4.3.3: Define coach assignment, capacity, duration, class credit cost.

R4.2: Coach Assignment & Notification

- R4.4.1: Assign coaches to classes; view schedule in coach mobile dashboard.
- R4.4.2: Trigger notifications on assignment or changes.

R5: Workout Design & Management (High Priority)

R5.1: Workout Creation for scheduled classes (Coach)

- R5.1.1: Free-text entry → structured templates (For Time, AMRAP, EMOM, Chipper, Intervals)
- R5.1.2: Tag workouts by category (strength, endurance, skill).

R5.2: Template Library

- R5.2.1: Save common WOD templates for reuse.
- R5.2.2: Create, delete and edit templates

R6: Payments & Subscriptions (High Priority)

R6.1: Payment Processing

- R6.1.1: Integrate recurring (monthly) and once-off payments
- R6.1.2: Enforce "pay-first" model: no booking without paid subscription/credit.
- R6.1.3: Generate and email receipts/invoices.

R6.2: Subscription & Packages

- R6.2.1: Define unlimited, fixed-credit, and custom-duration plans.
- R6.2.2: Auto-renewal with opt-out; configurable cancellation policies.
- R6.2.3: Member dashboard: view active plan, usage, expiry, and remaining credits.
- R6.2.4: Manager can define new packages

R7: Score Submission & Leaderboards (Medium Priority)

R7.1: Score Recording

- R7.1.1: Members submit their workout result (time, reps, rounds).
- R7.1.2: Coaches can review/edit scores post-class.
- R7.1.3: Coaches can fill in scores for members who cannot do so at the time.

R7.2: Leaderboards

- R7.2.1: Live daily leaderboard by class.
- R7.2.2: Overall (gym-wide) leaderboards.
- R7.2.3: Format-specific ranking rules (e.g., low-time vs. high-reps).
- R7.2.4: Privacy toggle: opt-out of public display.

R8: Reports & Analytics (Medium Priority)

R8.1: Attendance & Utilization

- R8.1.1: Class fill-rate, no-show rates, member attendance history.
- R8.1.2: Filterable by date range, class type, coach.

R8.2: Financial Metrics

- R8.2.1: Monthly revenue breakdown (recurring vs. one-off).
- R8.2.2: Overdue payments, churn rates.
- R8.2.3: Exportable CSV/PDF.

R8.3: Performance Trends

- R8.3.1: Popular workouts, attendance peaks.
- R8.3.2: Individual member improvement graphs and statistics.

R9: Communications & Notifications (Medium Priority)

R9.1: In-App Messaging

- R9.1.1: Coach ↔ member chat or class comment threads (optional).
- R9.1.2: Internal messaging between any user type.

R9.2: Push, Email & SMS Alerts

- R9.2.1: Booking confirmations, cancellations, reminders.
- R9.2.2: Payment receipts, renewal notices, expiry warnings.
- R9.2.3: Notification settings per user.
- R9.2.4: Alerts for coach substitutions and class detail changes. (mentioned earlier)

R10: Gamification & Engagement (Low Priority / Optional)

R10.1: Streaks & Badges

- R10.1.1: Track consecutive attendance streaks.
- R10.1.2: Points system or award badges for milestones (e.g., 50 WODs).

R10.2: Social Sharing

- R10.2.1: Share leaderboard placements or PRs to others.

R10.3: Avatar Evolution (Experimental)

- R10.3.1: Unlock avatar upgrades based on points; optional "avatar battle" mini-game.

R11: Workout Data Input Subsystem

All roles: Members (mobile), Coaches (mobile/web override)

R11.1: Common Input Framework

- R11.1.1: Real-time submission via WebSocket ensures the leaderboard updates instantly.
- R11.1.2: Allow manual entry/edit of the fields.

R11.2: "For Time" Workouts

R11.2.1: Timer Controls

- R11.2.1.1: "Start" button begins a stopwatch; timestamp recorded as start_time.
- R11.2.1.2: "Stop" button halts the stopwatch; timestamp recorded as end_time.
- R11.2.1.3: Display elapsed time in mm:ss format during timing.

R11.3: AMRAP (As Many Rounds/Reps As Possible)

R11.3.1: Time-cap and Countdown

- R11.3.1.1: Display the workout's time cap (e.g., "12:00") and a countdown timer.

R11.3.2: Rounds & Reps Entry

- R11.3.2.1: Two numeric fields: rounds_completed and extra_reps.
- R11.3.2.2: Real-time calculation of total_reps = rounds_completed × reps_per_round + extra_reps.

R11.4: EMOM (Every Minute on the Minute)

R11.4.1: Minute countdown

- R11.4.1.1: Specify how many minutes (intervals) in that workout
- R11.4.1.2: Display the countdown of the minute followed by short data entering period

R11.4.2: Number of intervals completed entry

- R11.4.2.1: Indicate whether they completed the task for that minute or not

---

R11.5: Chipper Workouts

R11.5.1: For-Time Flow

- R11.5.1.1: Identical UI to **R12.2**, since chipper is simply a long For Time.

R11.6: Interval Workouts (Fixed Work/Rest Blocks)

R11.6.1: Work/Rest Timer

- R11.6.1.1: Built-in timer alternates Work and Rest phases (configurable durations).
- R11.6.1.2: At end of each Work phase, prompt for reps_this_interval.

R11.7: Weight & Scaling Inputs

- R11.7.1: Reps and Load entry calculates score taking into account their body weight

---

User Stories

Title: Member Registration	**Priority: 1**	**Estimate:**
User Story: As a new member I want to register via mobile app using email and password So that I can access the gym services		
Acceptance Criteria: Given I am a new user on the registration page When I enter valid email, password and complete OTP verification Then my account is created and I can login		

Title: Coach login	**Priority: 1**	**Estimate:**
As a coach I want to login via mobile app using manager-provisioned credentials So that I can access coach features		
Acceptance Criteria: Given I have valid coach credentials When I enter them correctly in the login screen Then I am granted access to the coach sections of the app		

Title: Class Setup	**Priority: 1**	**Estimate:**
As a manager I want to create single or recurring classes So that I can manage the schedule		
Acceptance Criteria: Given I'm in the class management interface When I create a new class Then it appears in the schedule		

Title: Booking	**Priority: 1**	**Estimate:**
As a member I want to book classes using my subscription or credits So that I can attend sessions		
Acceptance Criteria: Given I have an active subscription credits When I book a class Then my credits are deducted and I receive a booking confirmation		

Title: Workout creation	**Priority: 3**	**Estimate:**
As a coach I want to create structured workouts for scheduled classes So that members can view and prepare for their workouts in advance		
Acceptance Criteria: Given I am assigned to a scheduled class When I create a workout Then it is saved and displayed for members attending that class		

Title: Leaderboard	**Priority: 2**	**Estimate:**
As a member I want to see workout leaderboards So that I can compare my performance with others		
Acceptance Criteria: Given I have submitted a workout score When I visit the leaderboard Then I see my score ranked alongside others from the same class		

Title: Leaderboard opt-out	**Priority:4**	**Estimate:**

As a member
I want to opt-out of my scores being on the leaderboard
So that others cannot see my score

Acceptance Criteria:

Given I have opted out of leaderboard visibility
When others view the leaderboard
Then my score does not appear in the rankings

Title: Manager Manages Roles	**Priority: 2**	**Estimate:**

As a manager
I want to create and revoke coach and admin accounts
So that only authorized users have access to sensitive features

Acceptance Criteria:

Given I am logged in as a manager
When I create a new coach or admin user
Then they receive credentials and can log in with role-based permissions

Title: Reminders	**Priority:4**	**Estimate:**

As a member
I want to receive reminders
So that I stay informed about bookings, cancellations, and changes

Acceptance Criteria:
Given I have a class booked
When the reminder time is reached
Then I receive a push or SMS notification

Title: Workout Submission	**Priority:1**	**Estimate:**
As a member I want to submit my workout result after class So that I can track my progress over time		
Acceptance Criteria: Given I attended a class When I open the score submission page Then I can enter my time, reps, or rounds		

Title: Member Profile	**Priority:2**	**Estimate:**
As a member I want to view and update my profile information So that my emergency contacts and details are current		
Acceptance Criteria: Given I am logged in When I navigate to the profile section Then I can view and edit my name, contact info, and emergency contacts		

Title: Coach Schedule	**Priority:1**	**Estimate:**
As a coach I want to see my assigned classes So that I can manage my time and prepare accordingly		
Acceptance Criteria: Given I am logged in as a coach When I open the schedule in my dashboard Then I see all my assigned classes with times		

Title: Search classes	**Priority:2**	**Estimate:**
As a member I want to search for classes and filter the search So that I can find classes that I want to attend		
Acceptance Criteria: Given there are classes that meet my criteria When I click search Then I can see all the classes that meet search parameters		

Title: Subscription management	**Priority:3**	**Estimate:**
As a manager I want to create new subscription packages So that I can create more offers for members		
Acceptance Criteria: Given I'm in the package management dashboard When I define a new package with unique parameters Then it becomes available for purchase		

Title: Subscription renewal	**Priority:3**	**Estimate:**
As a member I want to automatically receive payment receipts via email So that I have records for my purchases		
Acceptance Criteria: Given I've completed a payment When the payment processes successfully Then I receive a confirmation receipt via email		

Title: Member class payments	**Priority:4**	**Estimate:**
As a member I want to pay for a subscription So I can start booking classes		
Acceptance Criteria: Given I selected a plan When I complete a payment Then my access activates immediately		

Title: Opt-out of auto-renewal	**Priority:4**	**Estimate:**
As a member I want to be able to cancel auto-renewal of subscriptions So I can have full control of my payments		
Acceptance Criteria: Given I disable auto-renewal When I confirm the cancellation Then my plan expires on the end date		

User Characteristics

1. Members

Technical skill: Mostly average smartphone users; may not be tech-savvy.

Needs: Quick and intuitive class booking, easy navigation, reminders, and performance tracking.

Constraints: Limited time, low patience for long forms or slow apps.

Devices: Mobile-first (Android/iOS).

2. Coaches

Technical skill: Moderate—may be familiar with fitness apps or gym software.

Needs: View schedules, upload workouts quickly (text/media), access class rosters.

Constraints: Often working on tight schedules, need fast, responsive tools.

Devices: Mobile-first, some may use tablets.

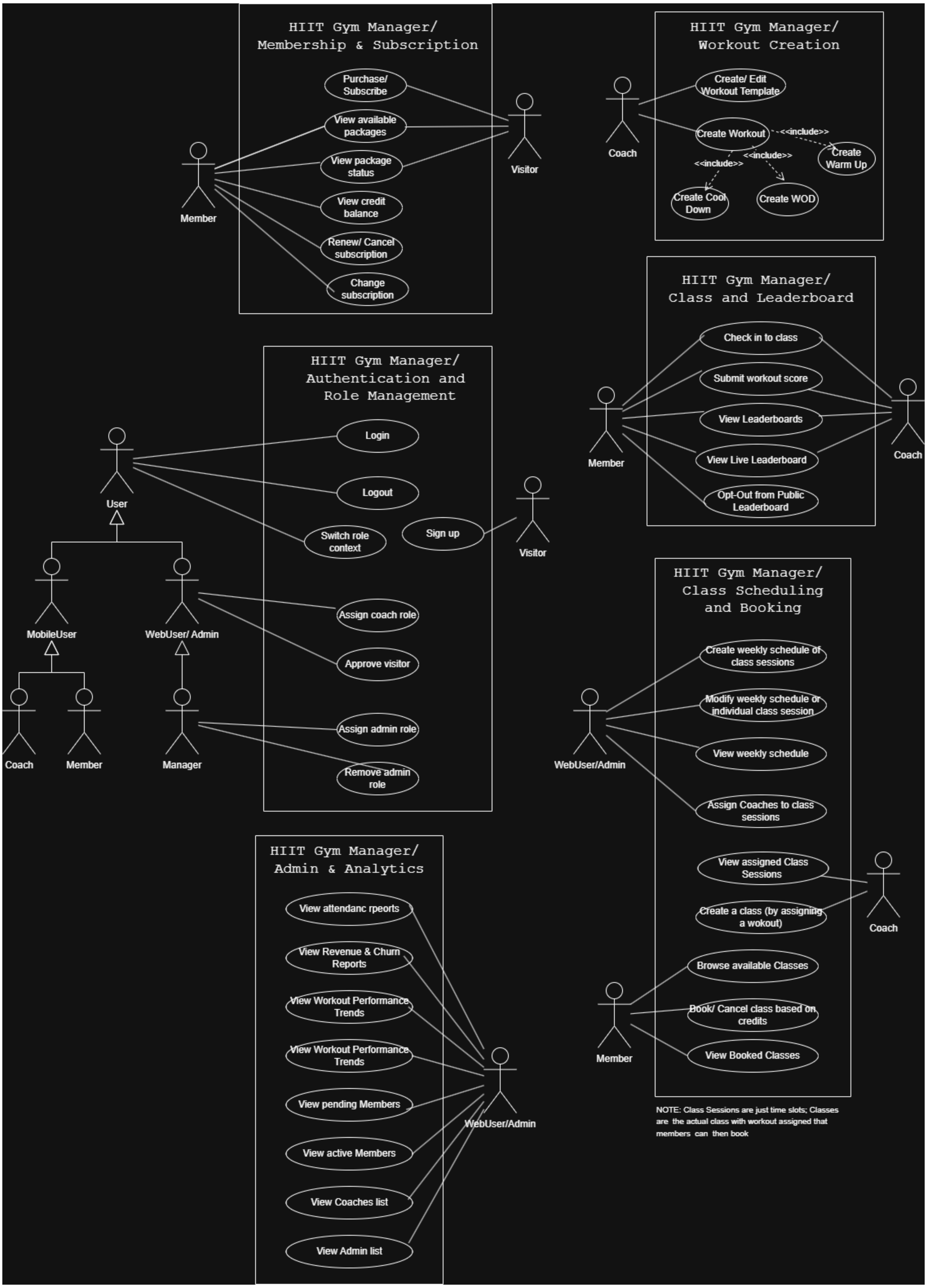
3. Managers/Admins

Technical skill: Moderate to high—may be familiar with spreadsheets, dashboards, and basic IT systems.

Needs: Oversee members and coaches, manage schedules, packages, reports.

Constraints: Need accuracy, stability, and some level of customization (e.g., setting packages or policies).

Devices: Desktop-focused, possibly using laptops or tablets.


HIIT Gym Manager/ Membership & Subscription
Purchase/ Subscribe
View available packages
View package status
View credit balance
Renew/ Cancel subscription
Change subscription
Member
Visitor
HIIT Gym Manager/ Workout Creation
Create/ Edit Workout Template
Create Workout
<<include>>
Create Warm Up
<<include>>
<<include>>
Create Cool Down
Create WOD
Coach
HIIT Gym Manager/ Class and Leaderboard
Check in to class
Submit workout score
View Leaderboards
View Live Leaderboard
Opt-Out from Public Leaderboard
Member
Coach
HIIT Gym Manager/ Authentication and Role Management
Login
Logout
Switch role context
Sign up
Assign coach role
Approve visitor
Assign admin role
Remove admin role
User
Visitor
MobileUser
WebUser/ Admin
Coach
Member
Manager
HIIT Gym Manager/ Class Scheduling and Booking
Create weekly schedule of class sessions
Modify weekly schedule or individual class session
View weekly schedule
Assign Coaches to class sessions
View assigned Class Sessions
Create a class (by assigning a wokout)
Browse available Classes
Book/ Cancel class based on credits
View Booked Classes
WebUser/Admin
Coach
Member
HIIT Gym Manager/ Admin & Analytics
View attendanc rpeorts
View Revenue & Churn Reports
View Workout Performance Trends
View Workout Performance Trends
View pending Members
View active Members
View Coaches list
View Admin list
WebUser/Admin

NOTE: Class Sessions are just time slots; Classes are the actual class with workout assigned that members can then book

Gym Management System: Service Contract Document

Group: Rome was built in a day
Project Name: Gym Manager

1. Authentication Service

Service Name	**Authentication Register**
Purpose	Allows users to register on the system
Input	JSON Body: { "firstname": string, "lastname": string, “email”:string, “phone” number, “password: string, “roles”: array }
Output	JSON: { "token": string, "user": { "userId": number, "role": string } }
Authentication	Not required
Consumers	Mobile App

Service Name	**Authentication Login**
Purpose	Authenticates users and issues JWT tokens for secure access
Input	JSON Body: { "email": string, "password": string }
Output	JSON: { "token": string, "user": { "userId": number, "role": string } }
Authentication	Not required
Consumers	Mobile/ Web App

2. Coach Class Services

a) View Assigned Classes

Service Name	**Coach Assigned Classes Service**
Purpose	Retrieve all classes assigned to the logged-in coach
Input	None (coach ID derived from JWT token)
Output	Array of classes: [{ classId, scheduledDate, scheduledTime, workoutId, coachId, capacity }, ...]
Authentication	Required
Consumers	Mobile App (Coach view)

b) Assign Workout to Class

Service Name	**Authentication Register**
Purpose	Assigns a workout to a scheduled class
Input	JSON: { "classId": number, "workoutId": number }
Output	{ "success": true }
Authentication	Required
Consumers	Mobile App (Coach view)

3. Member Booking Services

a) View available classes

Service Name	Member Available Classes Viewer
Purpose	Allows members to see what classes are available to book
Input	None (user ID derived from JWT)
Output	Array of bookings: [{ classId, scheduledDate, scheduledTime, workoutName }, ...]
Authentication	Required
Consumers	Mobile App (Member view)

b) View Booked Classes

Service Name	Member Booked Classes Viewer
Purpose	Allows members to see their class bookings
Input	None (user ID derived from JWT)
Output	Array of bookings: [{ bookingId, classId, scheduledDate, scheduledTime, workoutName }, ...]
Authentication	Required
Consumers	Mobile App (Member view)

c) Book a Class

Service Name	Class Booking Service
Purpose	Books a user into an available class
Input	JSON: { "classId": number }
Output	{ "success": true }
Authentication	Required
Consumers	Mobile App (Member view)

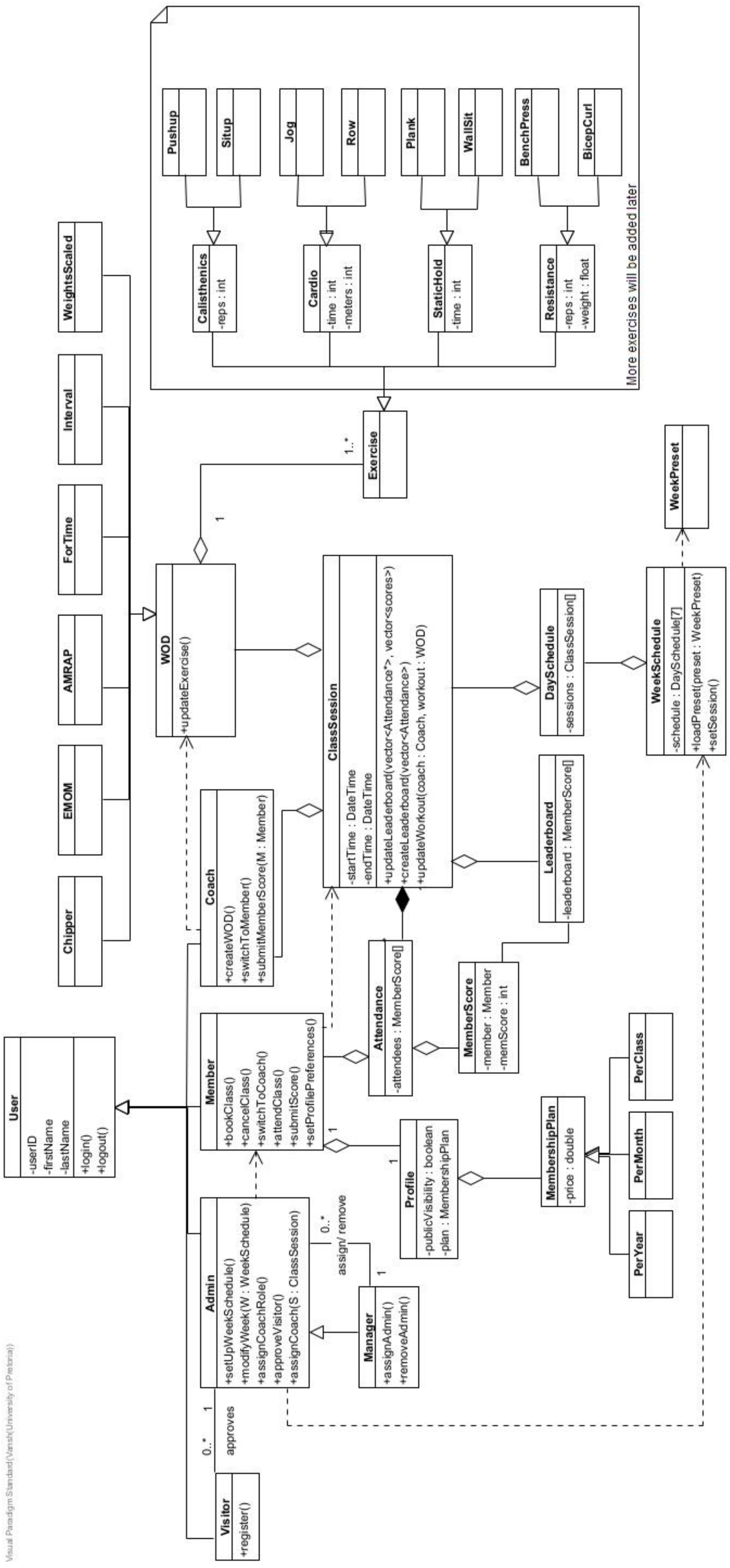
4. Scheduling Service (Manager)

a) Create a class

Service Name	Class Booking Service
Purpose	Books a user into an available class
Input	JSON: { "classId": number }
Output	{ "success": true }
Authentication	Required
Consumers	Mobile App (Member view)

c) Assign a coach to a class

Service Name	Class Booking Service
Purpose	Books a user into an available class
Input	JSON: { "classId": number }
Output	{ "success": true }
Authentication	Required
Consumers	Mobile App (Member view)



User
-userID
-firstName
-lastName
+login()
+logout()
Visitor
+register()
approves
0..*
1
Admin
+setUpWeekSchedule()
+modifyWeek(W : WeekSchedule)
+assignCoachRole()
+approveVisitor()
+assignCoach(S : ClassSession)
Manager
+assignAdmin()
+removeAdmin()
assign/ remove
0..*
1
Member
+bookClass()
+cancelClass()
+switchToCoach()
+attendClass()
+submitScore()
+setProfilePreferences()
Coach
+createWOD()
+switchToMember()
+submitMemberScore(M : Member)
Profile
-publicVisibility : boolean
-plan : MembershipPlan
1
1
MembershipPlan
-price : double
PerYear
PerMonth
PerClass
Attendance
-attendees : MemberScore[]
MemberScore
-member : Member
-memScore : int
Leaderboard
-leaderboard : MemberScore[]
ClassSession
-startTime : DateTime
-endTime : DateTime
+updateLeaderboard(vector<Attendance*>, vector<scores>)
+createLeaderboard(vector<Attendance>)
+updateWorkout(coach : Coach, workout : WOD)
DaySchedule
-sessions : ClassSession[]
WeekSchedule
-schedule : DaySchedule[7]
+loadPreset(preset : WeekPreset)
+setSession()
WeekPreset
WOD
+updateExercise()
1
1..*
Exercise
Chipper
EMOM
AMRAP
ForTime
Interval
WeightsScaled
Calisthenics
-reps : int
Cardio
-time : int
-meters : int
StaticHold
-time : int
Resistance
-reps : int
-weight : float
Pushup
Situp
Jog
Row
Plank
WallSit
BenchPress
BicepCurl
More exercises will be added later
Visual Paradigm Standard(Vansh(University of Pretoria))

Quality Requirements (Non-Functional)

- **QR1: Usability**
- **QR2: Security**
- **QR3: Reliability/ Availability**
- **QR4: Portability**
- **QR5: Maintainability**

SEE Architectural Requirements Document

For more details regarding the Quality Requirements and Architectures

Architectural Patterns

1. **Layered (n-Tier) Architecture - Overall**
2. **Microservices-Style Services - Backend**
3. **Event-Driven / Pub-Sub – Live & Notification System**
4. **Component-Based - Frontend**

Design Patterns

- **Pattern** Purpose & Location
- **Singleton** Shared DB connection pool and configuration loader
- **Factory / Builder** Create workout or notification objects based on format/type at runtime
- **Strategy** Scoring algorithms per workout format (For Time, AMRAP, EMOM, etc.)
- **Observer** Notification subsystem: subscribe to domain events to send emails/SMS
- **Adapter** Integrate external APIs (Stripe, Twilio, Setmore) behind uniform interface
- **Command** Encapsulate scheduled tasks (send reminders, generate reports)

Constraints

Technology Stack

- Frontend: Next.js (Web), React Native (Mobile)
- Backend: Node.js (API, workers, real-time)
- Database: PostgreSQL (with Drizzle ORM)
- Cache/Queue: Redis (or RabbitMQ)

Infrastructure & Deployment

- Monorepo managed by Turborepo with Yarn Workspaces or pnpm
- Containerization: Docker & Docker-Compose for local dev; Kubernetes for production
- CI/CD: GitHub Actions with lint/test/build pipelines

Compliance

- PCI-DSS for payment data (via Stripe's tokenization)
- GDPR-style data handling and digital consent storage

Operational

- Must support South Africa (Africa/Johannesburg) timezone scheduling
- Limit use of third-party vendor lock-in (use open standards where possible)

Resource

- Mobile app must operate on iOS 13+ and Android 9+
- Web UI optimized for modern browsers (Chrome, Safari, Edge)

Technology Requirements

1. Mobile App & Web Interface

- **Mobile Framework:** React Native (with Expo)
- **Web Framework:** Next.js
- **Language:** TypeScript
- **Rationale:**
  - Shared codebase → consistency and speed across platforms
  - Seamless state management using Redux
  - Cross-platform UI flexibility for roles-based interfaces

- **Alternatives:**

Option	Pros	Cons
Flutter	• Very fast UI • Single codebase like React Native	• Dart unfamiliar to team • Heavier APKs
Native Android + iOS	• Best performance • Full platform APIs	• Two codebases ⟶ doubles dev effort • Small team can't staff both
React Native CLI (bare)	• Same libs as Expo • More native control	• Build toolchain per OS ⟶ slower CI • OTA updates need extra infra

- **Decision:** React Native + Expo selected for its single-codebase maintainability and OTA availability gains within time/budget constraints.
- **Supports:**
  - QR – Maintainability (Single shared codebase)
  - QR – Availability (OTA hot-fix deploys)
- **Offline support:** Redux Persist (stores queued requests in SQLite)

2. Backend API & Business Logic

- **Runtime: Node.js (LTS)**
- **Framework: Express.js or NestJS**
- **Database: PostgreSQL and Drizzle ORM**
- **Rationale:**
  - Robust and scalable
  - Strong TypeScript support
  - Clear relational modelling for users, schedules, and tracking data

- **Alternatives:**

Option	Pros	Cons
NestJS	• Opinionated structure • Built-in DI, testing	• Heavier learning curve • Extra decorators/boilerplate
Django + DRF	• Admin site • Mature auth stack	• Python stack (new language) • Slower JSON throughput
Spring Boot	• Strong type-safety • Rich enterprise ecosystem	• Java verbosity • Higher memory footprint

- **Decision:** Express chosen for fastest feature velocity and direct alignment with the team's TypeScript expertise, while still meeting QR-Scalability (stateless, horizontal) and QR-Maintainability (simple, flat structure).
- **Supports:**
  - QR – Scalability (non-blocking I/O)
  - QR - Maintainability (lightweight, minimal layers)

---

3. Real-Time Features

- **Clock Sync & Live Leaderboards: WebSocket (Socket.IO)**
- **Live Workout Scoring & Judge Input: Socket.IO**
- **Rationale:**
  - Supports responsive, low-latency communication
  - Enables real-time scoring, notifications, and updates

- **Alternatives:**

Option	Pros	Cons
AWS Api Gateway (WebSocket)	• Fully managed, auto-scales • Native IAM auth possible	• Extra latency (region hop) • Monetary costs
Firebase Realtime Database	• Zero-config pub/sub • Built-in offline caching	• Data model = JSON tree - poor fit for relational scores • Vendor lock-in
pusher.com Channels	• Easy client SDKs • Webhooks for persistence	• Usage-based pricing • External service dependency

- **Decision:** Socket.IO aligns with the current TypeScript/Node skill set and stays entirely inside the project's code-base.

- **Supports:**
  - QR – Responsiveness (immediate push updates)
  - QR - Maintainability (single language and repo)

---

4. Data & Analytics

- **Analytics Dashboard: Custom dashboards with charting libraries (e.g., Recharts, D3.js)**
- **Usage Trends & Class Performance Metrics: Server-side aggregation via Node.js**

- **Alternatives:**

Option	Pros	Cons
Chart.js	• Simple API, • Good defaults	• Vanilla JS ⟶ custom React wrappers needed, • Less flexible for complex layouts
Apache ECharts	• Large chart library, • Handles huge datasets	• Heavier bundle size, • TypeScript typings less mature
Metabase (external BI)	• Auto-generated dashboards, • Non-dev users can explore	• Requires separate service and auth, • Limited custom styling

- **Decision:** Recharts selected: smallest learning curve for a React-native team and enough chart types for MVP dashboards, while server-side SQL keeps data logic in a single place.
- **Supports:**
  - QR – Maintainability (unified React codebase)
  - QR - Performance (push only aggregated rows to client)

---

5. Infrastructure & CI/CD

- **CI/CD: GitHub Actions + Terraform**
- **Rationale:**
  - GitHub Actions lives in the same repo ⟶ zero extra accounts, automatic PR pipelines.
  - Terraform keeps infra version-controlled and repeatable across dev machines.
- **Decision:** GitHub Actions + Terraform chosen for seamless GitHub integration, no vendor lock-in, and zero cost at MVP scale, while meeting QR-Maintainability (one-file pipeline) and QR-Availability (blue–green deploy jobs with instant rollback).
- **Supports:**
  - QR – Maintainability (pipeline-as-code in YAML)
  - QR - Traceability (IaC state versioned in repo)

- **Alternatives:**

Option	Pros	Cons
AWS Amplify (+ CloudFormation)	• Tight React support • Edge functions available	• Steeper AWS IAM setup • Costs can spike if mis-configured
Heroku Pipelines	• One-click deploys • Built-in rollback	• Free tier deprecated • Limited regions
GitLab CI + GitLab Runners	• Integrated container registry • Rich pipeline UI	• Entire repo must live on GitLab • Self-hosted runner setup for private projects

---

6. Testing & Quality

- Unit Tests: Jest (TS)
- Integration Tests: Jest (+ supertest)
- E2E (Mobile): Detox
- Linting: ESLint + Prettier
- **Rationale:**
    - Jest runs in the same Node/TypeScript runtime as the code → zero context-switch.
    - supertest lets us hit Express routes without a real network port.
    - Detox drives React-Native apps on device/simulator; fits our Expo stack.
    - ESLint + Prettier enforce consistent style automatically in GitHub Actions.
- **Decision:** We keep Jest + Detox because they align with the team's TypeScript/React-Native skillset, integrate smoothly with GitHub Actions, and incur no extra cloud cost or learning curve.
- **Alternatives:**
    - Mocha + Chai + Sinon or Vitest (Unit and Integration)
    - Appium or Expo E2E
    - StandardJS (Linting)
- **Supports:**
    - QR – Reliability (automated test suite on every PR)
    - QR – Maintainability (consistent code style gate in CI)

Why This Stack?

- Developer Efficiency: TypeScript-first stack enables rapid development with consistency across frontend and backend.
- User-Centric Design: Offline support and real-time updates cater to mobile-first gym users and staff.
- Cost-Effective Scalability: Uses free-tier and low-overhead tools while supporting rapid scale-up.
- Fitness-Tailored Features: Built-in leaderboard logic, rep counters, and workout tracking modules are tailored to HIIT routines.

Deployment Model (SUBJECT TO CHANGE)

Th HIIT Gym Management System will use a cloud-based deployment model to ensure accessibility, scalability, and ease of maintenance.

- The **frontend web app** (for managers/admins) will be deployed using **Vercel**.
- The **mobile app** (for coaches and members) will be distributed via **Expo**, accessible on both iOS and Android through QR or store deployment.
- The **backend API** and **PostgreSQL database** will be hosted on **Render** or **Railway**, with automated deployment pipelines triggered on push via CI/CD (GitHub Actions).
- The monorepo will be managed via **Turborepo**, enabling unified build and deployment workflows acrss web, mobile, and backend components.
- Environment-specific configurations (e.g. dev, staging, production) will be managed using .env files and platform-level secrets.

Live Deployed System (SUBJECT TO CHANGE)

A fully deployed and accessible version of the system will be available during the final demo:

- The **web admin interface** will be publicly accessible through a **Vercel-hosted URL**.
- The **mobile app** will be accessible via **Expo Go**, with support for QR code sharing or download from app stores (if time permits).
- A **live backend API** connected to a hosted PostgreSQL database will support real-time operations like booking, workout uploads, and leaderboard updates.
- All key user flows, including sign-up, login, class booking, and workout submission, will be testable on the live system.
- The deployment will include seeded demo data and distinct roles (e.g., coach, member, manager) to showcase end-to-end functionality during the live evaluation.