Ryder Cup Amateur Manager

A Full-Stack Web Application for Amateur Golf Tournament Management

Agustin Estevez Dominguez

v2.0.8 (BE) / v2.0.6 (FE) | February 2026

Frontend: github.com/agustinEDev/RyderCupWeb

Backend: github.com/agustinEDev/RyderCupAm

Agenda

1. Problem Statement

Challenge: Amateur golf tournaments still rely on spreadsheets, paper scorecards, and manual coordination. There is no accessible, modern platform tailored for Ryder Cup-format amateur events.

Pain Point	Required Solution
Manual scheduling	Automated round & match generation
Paper scorecards	Digital hole-by-hole score input
No live standings	Real-time leaderboard
Complex handicaps	WHS-compliant automatic calculations
Language barriers	Bilingual interface (EN/ES)
Security concerns	Enterprise-grade authentication

1.1 The Solution - Full-Stack Platform

70
API Endpoints

2,435
Total Tests

2
Languages

9.2-9.4
OWASP Score

Role	What they can do
Admin	Manage users, approve golf courses, system oversight
Creator	Create competitions, plan schedules, manage enrollments
Player	Browse & join tournaments, view schedules, input scores

Two repositories, one architecture: Both repos follow Clean Architecture + DDD with identical layered structure.

2. Full-Stack Architecture

```
┌─────────────────────────────────────────────────────────────┐
│                        FRONTEND                              │
│  React 19  |  Vite 7  |  Tailwind 4  |  1,104 tests         │
│  Clean Architecture  |  DDD  |  Zustand  |  i18n (EN/ES)    │
└──────────────────────────┬──────────────────────────────────┘
                           │  70 REST API calls
                           │  httpOnly cookies
                           │  Automatic token refresh
┌──────────────────────────┴──────────────────────────────────┐
│                        BACKEND                               │
│  FastAPI  |  Python 3.12  |  PostgreSQL 15  |  1,331 tests   │
│  Clean Architecture  |  DDD  |  RBAC  |  WHS Handicaps      │
└─────────────────────────────────────────────────────────────┘
```

Metric	Frontend	Backend
Architecture	Clean + DDD	Clean + DDD
Tests	1,104	1,331
Coverage	85%+	90%
OWASP	9.2/10	9.4/10

3. Technology Stack

Frontend

Technology	Role
React 19	UI framework
Vite 7.3	Build tool
Tailwind CSS 4	Styling
React Router 7	Navigation
Zustand 4	State management
Zod	Schema validation
Vitest 4	Unit testing
Playwright	E2E testing
Sentry 10	Error monitoring
react-i18next	i18n (12 namespaces)

Backend

Technology	Role
Python 3.12	Language
FastAPI 0.125	Web framework
PostgreSQL 15	Database
SQLAlchemy 2.0	ORM (imperative)
Alembic	DB migrations
pytest	Testing (parallel)
Sentry	APM + profiling
Docker	Containerization
Kubernetes	Orchestration
Mailgun	Email (ES/EN)

Part II

Backend

FastAPI + PostgreSQL + Clean Architecture

4. Backend Architecture

Clean Architecture - 3 Layers + Vertical Slicing

```
src/
├── modules/                # Vertical slices by domain
│   ├── user/               # User management module
│   │   ├── domain/         #   Entities, VOs, Repo interfaces
│   │   ├── application/    #   Use cases, DTOs, Mappers
│   │   └── infrastructure/ #   Routes, Persistence, Security
│   ├── golf_course/        # Golf course module (same layers)
│   ├── support/            # Support module (contact form)
│   └── competition/        # Competition + Schedule module
└── shared/                 # Cross-cutting concerns
    ├── domain/             #   Country entity
    ├── application/        #   Validators, Sanitizers
    └── infrastructure/     #   Middleware, DB, Email, Logging
```

330+
Source Files

61
Use Cases

13
Entities

37
ADRs

5. Domain Model & Database

Entities (13)

Module	Entities
User	User, PasswordHistory, RefreshToken, UserDevice
Golf Course	GolfCourse, Tee, Hole
Competition	Competition, Enrollment, Round, Match, TeamAssignment
Shared	Country

14 SQLAlchemy models with Alembic migrations
166 countries with 614 border relationships

Database (PostgreSQL 15)

```
users ──< user_devices
  |
  └──< enrollments >── competitions
                            |
                      ├──< rounds
                      |      └──< matches
                      └──< team_assignments


golf_courses ──< tees
     └──< holes


countries ──< country_adjacencies
```

Full ERD documented in `docs/DATABASE_ERD.md` (Mermaid)

6. API Design - 70 REST Endpoints

Module	Endpoints	Scope
Auth	11	Login, register, verify, refresh
Users	4	Profile, security, roles
Devices	2	Fingerprinting, revocation
Handicaps	3	Manual + RFEG
Golf Courses	10	CRUD, approval workflow
Competitions	10	CRUD, state machine
Enrollments	8	Request, approve, reject
Schedule	11	Rounds, matches, teams
Support	1	Contact form → GitHub
Countries	2	List, adjacent
Total	**70**	

Documentation

- **Swagger UI:** Auto-generated from FastAPI
- **ReDoc:** Alternative API browser
- `docs/API.md`**:** Full endpoint reference
- **ADR-036:** SBOM REST API design

Key Design Decisions

- RESTful resource naming
- Consistent error responses (RFC 7807)
- Pagination on list endpoints
- Rate limiting per endpoint (5-100 req/min)

7. Backend Security & DevOps

Security (OWASP 9.4/10)

Feature	Implementation
Auth	httpOnly cookies + JWT
Password	bcrypt + history (last 5)
Lockout	10 attempts, 30min unlock
CSRF	Triple-layer protection
Rate limit	SlowAPI per-endpoint
Headers	HSTS, CSP, X-Frame
Audit	8 event types, JSON logs
Devices	SHA256 fingerprinting

DevOps & CI/CD

Component	Technology
Container	Docker
Orchestration	Kubernetes (Kind)
CI/CD	GitHub Actions (10 jobs)
Hosting	Render.com
Monitoring	Sentry (APM + profiling)
Email	Mailgun (EU, ES/EN)

Pipeline: Lint (Ruff) > Type check (mypy) > Security scan > Tests > Coverage > Docker build > Deploy

7.1 Backend Documentation

37 Architecture Decision Records (ADRs):

ADR	Topic
001	Clean Architecture
002	Value Objects
005	Repository Pattern
006	Unit of Work
007	Domain Events
012	Composition Root
020	Competition Domain
026	WHS Handicap Calc

ADR	Topic
023	OWASP Compliance
027	Account Lockout
028	CSRF Protection
030	Device Fingerprinting
031	Match Play Scoring
032	Golf Course Approval
033	Invitation Tokens
037	Handicap Session Model

Plus: Database ERD, Threat Model, Runbook, Module docs, Security docs, CI/CD docs

Part III

Frontend

React 19 + Vite + Clean Architecture

8. Frontend Architecture

Clean Architecture - 4 Layers

```
┌─────────────────────────────────────────────────────┐
│ PRESENTATION    23 Pages | 35+ Components | Hooks   │
├─────────────────────────────────────────────────────┤
│ APPLICATION     59 Use Cases                        │
├─────────────────────────────────────────────────────┤
│ DOMAIN          9 Entities | 21 VOs | Interfaces    │
├─────────────────────────────────────────────────────┤
│ INFRASTRUCTURE  12 API Repos | Mappers (ACL)        │
└─────────────────────────────────────────────────────┘
           Dependencies point INWARD only
```

280+
Source Files

~43K
Lines of Code

59
Use Cases

21
Value Objects

8.1 Anti-Corruption Layer

Mapper Pattern - Isolating Domain from API

```
Backend API (Python)
─────────────────

competition_id
session_type
team_a_players: [{
  user_id: "...",
  playing_handicap: 12
}]
```

```
Frontend Domain (JS)
────────────────────

competitionId
sessionType
teamAPlayers: [{
  userId: "...",
  playingHandicap: 12
}]
```

Composition Root (`src/composition/index.js`) wires all 59 use cases with their 12 repositories at startup, keeping all layers fully decoupled. Domain layer has zero external dependencies.

9. Key Features - Tournament Lifecycle

```
CREATE  ──>  ENROLL  ──>  SCHEDULE  ──>  PLAY  ──>  LEADERBOARD
```

Phase	Key Actions
Create	Team names, play mode, golf course
Enroll	Tee category, handicap, approval
Schedule	Auto/manual match generation
Play	Hole-by-hole scoring, walkover
Results	Real-time leaderboard, polling

Match Format	Players
Singles	1 vs 1
Fourball	2 vs 2 (best ball)
Foursomes	2 vs 2 (alternate shot)

Handicap Modes:

- Stroke Play / Match Play
- Allowance: 50-100% (5% steps)
- WHS formula: `PH = (HI x SR/113) + (CR - Par)`

9.1 Schedule Management

Round Configuration

- Date + session (Morning/Afternoon/Evening)
- Golf course selection
- Match format (Singles/Fourball/Foursomes)
- Handicap mode + allowance %

Team Assignment

- Automatic (balanced by handicap)
- Manual (drag & drop with @dnd-kit)

Match Generation

- Automatic player pairing
- WHS handicap calculations
- Strokes given per hole
- Walkover declaration with reason

UI Components (8 new)

- RoundCard, MatchCard, MatchDetailModal
- TeamAssignmentSection, AssignTeamsModal
- WalkoverModal, ReassignPlayersModal
- EnrollmentRequestModal

10. Frontend Security & i18n

Security (OWASP 9.2/10)

Feature	Detail
Tokens	**httpOnly cookies**
Access	15 min lifetime
Refresh	7 days, auto-rotation
Validation	HTML + Zod + Pydantic
Logout	Multi-tab (Broadcast API)
Assets	SRI hashes
Monitoring	Sentry + session replay

Internationalization (EN/ES)

Namespace	Scope
`auth`	Login, register, sessions
`common`	Header, footer, shared
`competitions`	Tournaments, enrollment
`schedule`	Rounds, matches, teams
`golfCourses`	CRUD, approval
`profile`	User profile
`dashboard`	Dashboard
`devices`	Device management
`landing`	Landing page
`pricing`	Pricing plans
`contact`	Contact form
`legal`	Terms, privacy, cookies

Auto-detection + localStorage + flags

11. Testing Strategy

1,331	1,104	90%	85%+
Backend Tests	Frontend Tests	Backend Coverage	Frontend Coverage

Backend (pytest)

Layer	Focus
Domain	Entity invariants, VO rules
Application	Use case orchestration
Infrastructure	Repository, API routes
Integration	Full endpoint flows

Parallel execution with `pytest-xdist`

Frontend (Vitest)

Layer	Files
Domain	28
Application	38
Infrastructure	11
Hooks/Utils	6

Philosophy: Mock at repo boundaries, never mock HTTP directly

12. CI/CD Pipelines

Backend (10 jobs, ~3 min)

```
Lint (Ruff)
  └─> Type Check (mypy)
    └─> Security Scan
      └─> Tests (parallel)
        └─> Coverage (>=80%)
          └─> SAST (CodeQL)
            └─> Docker Build
              └─> Deploy
```

- License audit + Snyk + pip-audit

Frontend (3 workflows)

```
Lint (ESLint 9)
  └─> Tests (Vitest, 1104)
    └─> Coverage (>=85%)
      └─> Build (Vite)
        └─> Bundle budget (<=1400 KB)
          └─> Deploy (Vercel)
```

- Security scan (TruffleHog, npm audit)
- PR checks (size, conventional commits)

Both repos: GPG-signed commits required, branch protection on `main`

13. Roadmap - Upcoming Sprints

Sprint 3: Invitations (Feb 25 - Mar 3)

- Invitation cards (accept/decline UI)
- Email-based invitations by user ID or email
- Invitation status tracking and badges
- 5 new endpoints (secure tokens, auto-enrollment)

Sprint 4: Live Scoring (Mar 4 - Mar 17)

- Scoring page with **3 tabs**: Input, Scorecard, Leaderboard
- Hole-by-hole score input with real-time validation
- Dual validation (player + marker)
- Scorecard submission workflow
- Polling every 10s for live updates

Sprint 5: Leaderboard (Mar 18 - Mar 24)

- Public leaderboard page (no auth required)
- Team standings bar (aggregate scores)
- Match summary cards with results
- Redis cache + conditional polling (30s)

v2.1.0: GDPR + Audit + Avatars

- GDPR: Data export (JSON), account deletion, consent logging
- Audit trail with DB persistence + CSV export
- Avatar system (Cloudinary/S3, max 2 MB)

v2.2.0: AI & RAG Module

- Golf rules assistant chatbot (see next slide)

14. AI & RAG - Golf Rules Assistant (v2.2.0)

Architecture

```
src/modules/ai/
├── domain/           # Entities, VOs, Interfaces
├── application/      # Use Cases, DTOs, Ports
└── infrastructure/   # Pinecone, Redis, OpenAI
```

Stack: LangChain + Pinecone + GPT-4o-mini
Cost: ~$1-2/month
Knowledge Base: R&A Official Rules of Golf

Endpoints:

- `POST /competitions/{id}/ai/ask`
- `GET /competitions/{id}/ai/quota`

Key Design Decisions

Feature	Detail
Availability	Only during `IN_PROGRESS` competitions
Rate limits	10/day global, 3/day player, 6/day creator
Cache	Redis TTL 7 days (80% hit rate expected)
Pre-FAQs	20-30 hardcoded common questions
Temperature	0.3 (factual, low creativity)
Tests	60+ tests (mocking OpenAI)

Ports (Clean Architecture):
VectorRepository, CacheService, DailyQuotaService, LLMService

> **RAG ensures accurate, citation-based answers** from the official golf rulebook, not hallucinated responses.

15. Lessons Learned

Architecture

Clean Architecture pays off
API response changed from flat to nested arrays - only the mapper layer needed updates. Zero domain changes.

Shared architecture enables collaboration
Both repos using Clean + DDD means shared vocabulary and patterns across the full stack.

37 ADRs document every decision
Architecture Decision Records provide audit trail and onboarding material.

Process

i18n from day one
Retrofitting 30+ toast messages was far harder than building with translations from the start.

Test the domain, not the UI
Domain tests are stable across refactors. UI tests break on style changes.

httpOnly cookies > localStorage
Eliminates XSS token theft entirely. Worth the extra CSRF handling.

15.1 Technical Decisions

Decision	Frontend	Backend
Architecture	Clean + DDD (4 layers)	Clean + DDD (3 layers + vertical)
State	Zustand	SQLAlchemy UoW
HTTP	Fetch API	FastAPI async
Validation	Zod schemas	Pydantic models
Auth	httpOnly cookies	JWT + refresh tokens
Testing	Vitest (1,104)	pytest (1,331)
Monitoring	Sentry (errors)	Sentry (APM + profiling)
CI/CD	GitHub Actions (3 workflows)	GitHub Actions (10 jobs)
Docs	11 ADRs + API spec	37 ADRs + ERD + Runbook
Deploy	Vercel	Docker + Kubernetes + Render

Thank You

Ryder Cup Amateur Manager

Frontend: github.com/agustinEDev/RyderCupWeb
Backend: github.com/agustinEDev/RyderCupAm

2,435	**9.2 / 9.4**	**70**	**48**
Tests (1,104 FE + 1,331 BE)	OWASP Score (FE / BE)	REST API Endpoints	ADRs (11 FE + 37 BE)