**Project Design Documentation Template:**

Last updated: August 14th, 2025

1. **Project Title & Version Control**

Project Title: ScoutConnect – Smart Scouting Platform for Talent Discovery & Collaboration

*Version Control*

Version: v1.0

Date: 08/14/2025

Change Log: Initial comprehensive design document based on 12-week engineering plan

1. **Project Summary**

ScoutConnect is a backend-driven sports scouting platform that centralizes player data, provides intelligent evaluation tools, and enables collaboration between scouts and coaches. The system streamlines athlete discovery by using standardized criteria, scoring algorithms, and searchable player profiles to highlight both visible and hidden talent. Built with FastAPI and SQLite, it provides a secure, scalable foundation for sports talent management with JWT authentication, role-based access control, and comprehensive testing coverage.

What is your project about and why does it matter?

**3. Problem Statement / Use Case**

Many skilled athletes remain unnoticed due to limited exposure, inconsistent scouting standards, and fragmented data storage across different organizations. Current scouting methods lack standardization, making it difficult to compare players fairly or identify hidden talent. ScoutConnect addresses this by providing a unified backend platform where coaches, scouts, and sports organizations can store, analyze, and share player performance metrics using consistent evaluation criteria. The platform ensures fair evaluation, faster decision-making, and greater opportunities for under-the-radar athletes across multiple sports including lacrosse, basketball, and football.

**Primary Users:**

* Coaches seeking to evaluate track player performance
* Scouts looking to discover and compare talent
* Sports administrators managing player databases
* Athletic organizations needing standardized evaluation systems

What real-world problem does this solve? Who would use this?

**4. Goals and Objectives**

* Build a secure, scalable backend API with JWT authentication, comprehensive CRUD operations, and role-based permissions (Coach, Scout, Admin)
* Develop intelligent scoring systems with both universal metrics (Speed, IQ, Clutch, Strength, Effort) and sport-specific evaluation criteria with customizable weightings
* Enable seamless collaboration through watchlists, comments, evaluation sharing, tagging systems, and player comparison tools
* Implement data-driven talent discovery with recruitability algorithms, hidden gem detection, and injury risk prediction

List 2--3 clear and specific technical goals of your project.

**5. Key Features / Functions**

**Authentication & Security**

* JWT-based authentication with secure password hashing using passlib
* Role-based access control (Coach, Scout, Admin) with appropriate permissions
* Secure API endpoints with comprehensive input validation

**Player Management**

* Comprehensive player profiles with detailed performance evaluations
* CRUD operations for players with standardized data storage
* Multi-sport support with sport-specific evaluation criteria

**Scoring & Analytics**

* Universal scoring engine with metrics: Speed (0-10), IQ (0-10), Clutch (0-10), Strength (0-10), Effort (0-10)
* Sport-specific scoring formulas and weightings
* Growth tracking and performance trend analysis
* Clutch performance calculation based on high-pressure situations

**Talent Discovery**

* Recruitability assessment based on growth trends, injury history, clutch rating, effort, and coach feedback
* Hidden gems detection algorithm for high-potential, low-visibility players
* Advanced filtering by age, sport, competition level

**Collaboration Tools**

* Personal watchlists for tracking interesting players
* Player comparison tools for side-by-side metric analysis
* Comment system for evaluations and player profiles
* Tagging system (#underrated, #clutch, etc.) for easy categorization
* Evaluation sharing between authorized users

**Data Integration**

* CSV/JSON stat upload functionality for bulk data import
* Real-time stat integration with automatic score recalculation
* Game-by-game performance tracking (last 5 games)

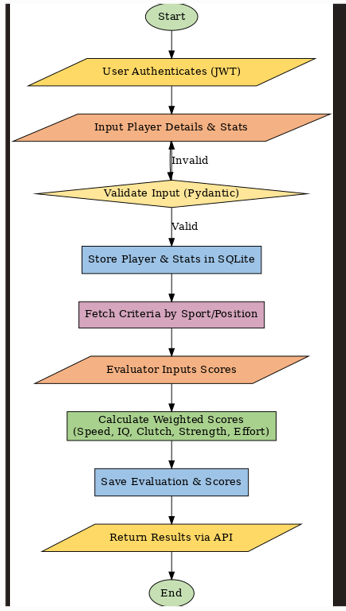
**Health & Risk Management**

* Injury tracking and recovery monitoring
* Injury risk prediction based on load, performance drops, and injury history
* Comprehensive health summaries per player

What are the main things your project will do?

**6. Tech Stack and Tools**

**Backend Framework:** FastAPI (Python)  
**Database:** SQLite with SQLAlchemy ORM  
**Authentication:** JWT with passlib for password hashing  
**Testing:** pytest with comprehensive unit and integration tests  
**API Documentation:** Swagger/OpenAPI (auto-generated)  
**Development Tools:** uvicorn (ASGI server), Pydantic (data validation)  
**Version Control:** Git with GitHub repository  
**Containerization:** Docker for deployment  
**Dependencies:** requests, python-dotenv for environment management



Languages, frameworks, platforms, or security tools you'll use (e.g., Python, Flask, PostgreSQL, Zeek, Wireshark)

**7. Architecture / Workflow Diagram**

**Core Database Schema:**

* **users**: Authentication and role management
* **players**: Core player profiles and basic information
* **evaluations**: Performance assessments linked to players and evaluators
* **stats**: Game statistics and performance metrics
* **criteria\_templates**: Custom evaluation criteria per sport/position
* **tags**: Categorization system for players
* **player\_tags**: Many-to-many relationship for player categorization
* **watchlist**: Personal player tracking for users
* **comments**: Collaboration and feedback system
* **injuries**: Injury tracking and recovery monitoring

**API Endpoint Structure:**

/auth (registration, login, logout)

/players (CRUD operations)

/evaluations (performance assessments)

/criteria-generator (sport-specific evaluation forms)

/scoring (player score calculations)

/recruitability (talent assessment)

/hidden-gems (undervalued player detection)

/compare (player comparison tools)

/watchlist (personal tracking)

/comments (collaboration)

/stats/upload (data ingestion)

/injury-report (health monitoring)

Algorithm and Flowchart for project

**8. Timeline / Weekly Milestones**

Week Outcome

Week 1: Project Kickoff & GitHub repo, virtual environment, Environment Setup dependencies installed, basic FastAPI "Hello World" running

Week 2: Database Design + SQLite DB with SQLAlchemy, complete User Authentication ERD implementation, JWT auth with role-based access control

Week 3: Core API: Player /players and /evaluations CRUD Profiles & Evaluations routes, Pydantic validation, comprehensive unit tests

Week 4: Multi-Sport Criteria /criteria-generator endpoint, System sport-specific templates, custom criteria storage and testing

Week 5: Scoring Engine Universal and sport-specific Implementation scoring formulas, /player/{id}/score endpoint with breakdowns

Week 6: Recruitability & /recruitability and /hidden-gems Hidden Talent Engine endpoints with filtering capabilities and scenario testing

Week 7: Player Comparison + /compare and /watchlist endpoints Watchlists with role-based testing and functionality validation

Week 8: Collaboration & /comments and /share-eval endpoints, Comments tagging system implementation and testing

Week 9: Real-Time Stat /stats/upload endpoint, CSV/JSON Integration parsing, automatic score recalculation, trend tracking

Week 10: Injury Risk & Injuries table, risk prediction Recovery Tracker algorithms, /injury-report endpoint with simulated data testing

Week 11: Testing, Validation & Comprehensive unit/integration tests, Error Handling input validation, error handling, automated API documentation

**Week 12: Final Polish & Code refactoring, documentation Demo Prep completion, demo video creation, presentation preparation**

**9. Risks and Risk Mitigation**

**Complex scoring algorithms may delay development** --- mitigate with modular implementation and early testing, start with basic formulas and iterate to avoid blocking core functionality.

**Database structure changes mid-project** --- mitigate with thorough initial ERD design and SQLAlchemy migration tools, comprehensive schema documentation to prevent major refactoring.

**Authentication and security implementation complexity** --- mitigate by using proven libraries (passlib, JWT), implement early with comprehensive testing, follow established security best practices.

**Time constraints and feature creep** --- mitigate with disciplined adherence to milestones and prioritization of core features, maintain detailed project tracking and avoid scope expansion.

**Integration complexity with multiple data sources** --- mitigate with robust error handling, comprehensive input validation, and extensive testing with mock CSV/JSON data scenarios.

What risks are there to successful completion of the project and what can you implement to mitigate the impact of those risks.

**10. Evaluation Criteria**

**All API endpoints functional, secure, and properly tested** with pytest ensuring comprehensive coverage of CRUD operations, authentication, and role-based permissions across all routes.

**Database correctly stores, retrieves, and updates all required data** with proper relationships between users, players, evaluations, stats, and all supporting tables maintaining data integrity.

**Comprehensive API documentation auto-generated** with Swagger/OpenAPI providing clear endpoint descriptions, request/response schemas, and authentication requirements for easy integration.

**Scoring and recruitability algorithms produce accurate, reproducible results** with proper mathematical formulas implemented for Speed, IQ, Clutch, Strength, Effort metrics and hidden talent detection.

**All collaboration features fully implemented and tested** including watchlists, comments, tagging system, and evaluation sharing working correctly across different user roles.

**Project deployable with Docker containerization** and complete README with setup instructions, usage examples, and successful demo presentation showcasing key features.

How will you know this is done well? List at least 3 measures of success.

**11. Future Considerations**

**Integration with live sports data APIs** (ESPN, sports databases) for real-time stat updates and automatic player data synchronization to reduce manual data entry.

**Mobile app integration** with REST API consumption for scouts and coaches providing field-ready evaluation tools and real-time player assessment capabilities.

**Machine learning models** to predict player potential and career trajectories using historical performance data, injury patterns, and growth metrics for advanced talent identification.

**Enhanced security features** with two-factor authentication, advanced activity logging, and audit trails for sensitive player data and evaluation access.

**Multi-tenant architecture** for supporting multiple organizations with isolated data, custom branding, and organization-specific evaluation criteria and workflows.

**Advanced analytics dashboard** with data visualization, reporting tools, and trend analysis for comprehensive performance insights and decision-making support.

What long term maintenance or added functionality will be needed in the future