12-Week Backend Engineering Plan (SQLite Edition)

Project: ScoutConnect – Smart Scouting Platform for Talent Discovery & Collaboration
Stack: FastAPI + SQLite + Pytest + JWT Auth

PHASE 1 – Foundation & Setup (Weeks 1–4)

Week 1 – Project Kickoff & Environment Setup

- Create GitHub repository & initialize project.
- Set up **virtual environment** & install dependencies:
    - `fastapi`, `uvicorn`, `sqlite3`, `pydantic`, `sqlalchemy`, `pytest`, `requests`, `passlib`.

Create folder structure:

bash
CopyEdit

```
/app
  /models
  /routes
  /services
  /utils
/tests
```

-
- Configure `.env` for secret keys & settings.
- Add `.gitignore`.
- Add **README.md** with project overview & tech stack.

- Run a sample FastAPI “Hello World” to confirm setup.

Week 2 – Database Design + User Authentication

- Implement **SQLite** DB using **SQLAlchemy ORM**.
- Create **ERD** based on robust schema:
    - `users`, `players`, `evaluations`, `stats`, `tags`, `player_tags`, `watchlist`.
- Implement **user registration/login/logout** with JWT.
- Hash passwords with `passlib`.
- Enforce role-based access control (`Coach`, `Scout`, `Admin`).
- Write unit tests for auth routes using `pytest` + `requests`.

Week 3 – Core API: Player Profiles & Evaluations

- Build `/players` CRUD routes.
- Build `/evaluations` routes:
    - Create evaluation linked to player & evaluator.
    - Retrieve evaluations by player ID.
- Add **input validation** with Pydantic:
    - Speed, IQ, Clutch, Strength, Effort all 0–10 (float).
- Unit test player & evaluation endpoints.

Week 4 – Multi-Sport Criteria System

- Build `/criteria-generator` endpoint:
    - **Input:** `sport`, `position`.
    - **Output:** Predefined evaluation form fields with weightings.
- Implement default criteria per sport (e.g., lacrosse, basketball, football).
- Allow coaches to save **custom criteria templates**.
- Store templates in DB table `criteria_templates`.
- Test with automated Python tests, not Postman.

PHASE 2 – Smart Logic & Collaboration (Weeks 5–8)

Week 5 – Scoring Engine (Universal + Sport-Specific)

- Implement **scoring formulas**:
    - **Speed:** `(sprint_baseline / player_sprint_time) * 10`
    - **IQ:** `(correct_decisions / total_decisions) * 10`
    - **Clutch:** `(performance_last_quarter / average_performance) * 10`
    - **Strength:** `(bench_press / max_bench) * 10`
    - **Effort:** `((minutes_played / total_minutes) + hustle_ratio) / 2 * 10`
- Build `/player/{id}/score` endpoint with breakdown:
    - Overall score, clutch %, growth %.
- Write automated scoring tests with realistic mock data.

Week 6 – Recruitability & Hidden Talent Engine

- `/recruitability` endpoint:
    - Factors: growth trend, injury history, clutch rating, effort, coach feedback.
- `/hidden-gems` endpoint:
    - Players with high growth/effort but low visibility.
- Add filters: age, sport, level.
- Test with various dataset scenarios in Python.

Week 7 – Player Comparison + Watchlists

- `/compare` endpoint:
    - Compare 2+ players side-by-side on metrics & scores.
- `/watchlist` endpoint:
    - Add/remove players to personal watchlist.
    - View all watchlisted players per user.
- Write watchlist tests for different roles (coach/scout).

Week 8 – Collaboration & Comments

- `/comments` endpoint: Add/view comments on players/evaluations.
- `/share-eval` endpoint: Share evaluation with other users (via DB link, not email).
- Implement tagging system (`#underrated`, `#clutch`).

- Store tags in `tags` table, link via `player_tags`.
- Test collaboration features.

PHASE 3 – Data Ingestion, Testing & Final Polish (Weeks 9–12)

Week 9 – Real-Time Stat Integration

- `/stats/upload` endpoint:
    - Accept CSV/JSON with game stats.
    - Update `stats` table & recalculate scores.
- Track last 5 games, game-by-game trends.
- Link clutch rating recalculation to recent games.
- Test CSV parsing & data update flow.

Week 10 – Injury Risk & Recovery Tracker

- Add `injuries` table:
    - Player ID, injury type, date, recovery notes.
- Predict injury risk:
    - Based on load, stat drops, past injuries.
- `/injury-report` endpoint: Return health summary per player.
- Test with simulated injury histories.

Week 11 – Testing, Validation & Error Handling

- Add **full unit + integration tests** (pytest).
- Validate all inputs (Pydantic).
- Handle: auth errors, 404s, invalid JSON, duplicates.
- Generate **automatic API docs** with Swagger (FastAPI's `/docs`).

Week 12 – Final Polish & Demo Prep

- Refactor code & add docstrings.
- Update README with:
    - Setup guide
    - API usage examples
    - Testing instructions
- Record a **demo video** walking through features.
- Present to bootcamp/mentors.

Deliverables

✅ Fully functional FastAPI backend with SQLite DB.
✅ JWT-based auth & role permissions.
✅ Player scoring, watchlists, comparison, tags.
✅ Automated tests (no Postman).
✅ Dockerized local deployment.