# **Project Plan – DeepChessIQ (10+2 Weeks)1. Scope Statement**

* **Phase I Features (MVP)**
  + Accounts, PGN import/export, storage.
  + RL Bot (Stockfish baseline + adjustable personality).
  + LLM-based game analysis summaries.
  + Tactics trainer based on weaknesses.
  + Commentator QuickPack.
  + OBS live overlay.
  + Coach dashboard.
* **Technical Deliverables**
  + Backend microservices, frontend web app, real-time overlays, integrations, deployment.

## **2. Requirements & Deliverables**

### ***Functional Requirements***

* User accounts, PGN import/export.
* AI RL bot with adjustable play styles.
* LLM-based plain-language analysis of games.
* Tactics trainer with spaced repetition.
* Real-time overlays for commentators.
* Basic coach dashboard with assignments.

### ***Non-Functional Requirements***

* **Performance**: Handle up to 100 concurrent analysis jobs.
* **Scalability**: Modular backend to add Phase II features easily.
* **Usability**: Clean UI, coach/player separation.
* **Security**: OAuth integration with Chess.com/Lichess, secure storage.

### ***End Product Expectations***

* *Working* ***SaaS MVP*** *deployed by* ***Week 12****.*
* *Fully tested, demo-ready product.*
* *Documentation (user + developer).*
* *Progress reporting aligned with activity log.*

**3. Work Schedule & Timeline**

### **Key Milestones**

* **Week 4:** Core architecture + baseline features working.
* **Week 7:** Full feature integration complete.
* **Week 10:** MVP build frozen (feature complete).
* **Weeks 11–12:** QA, stress testing, documentation, deployment.

# **10 Week Build Plan**

# **Week 4**

**AI:** Build chess AI foundations (basic, not full RL yet).

* **Stockfish Integration:** Wrap Stockfish (Python + UCI) → expose service (input FEN/PGN → output eval + best move).
* **Dataset Prep:** Collect & clean PGNs (Lichess/Chess.com), organize by rating bands, store in Postgres/S3.

**Deliverable:** Stockfish wrapper service + cleaned PGN dataset.

**SW: Set up backend + frontend skeletons.**

* **Backend:** Auth service (JWT), Postgres schema (Users, Games), PGN API (upload/download).
* **Frontend:** Init React project, router (/login, /dashboard, /coach), wireframes for dashboards, component library.

## **Week 5**

**AI:** LLM-based Game Summaries (Prompt + Caching)

* Build pipeline for converting engine outputs (Stockfish evals) into natural language using an LLM.
* Design prompt template: *position summary, key moments, plan suggestions*.
* **Implement retrieval + caching layer:**
  + Cache engine analysis to avoid re-querying LLM for the same position.
  + Store outputs in DB for re-use.
* Test with sample PGNs (short games, different ratings).

**SW: PGN Upload/Download + Cloud Storage**

* **Build API endpoints for:**
  + Upload PGN → store in Postgres/S3.
  + Download PGN → retrieve from storage.
* Implement file validation (reject corrupted PGNs).
* Connect frontend: UI for “Upload Game” + “Download PGN”.
* Deliver a basic storage layer with user mapping (games linked to users).

**Deliverable:**

* **AI**: First LLM-generated game summaries.
* **SW**: Upload/download PGN feature working end-to-end.

## **Week 6**

**AI:** Tactics Trainer MVP

* Extract tactical motifs from user games (blunders, missed wins, hanging pieces).
* **Generate puzzles:** “Find best move” positions from real games.
* Apply spaced repetition logic (review weaknesses more frequently).
* Store puzzle sets in DB per user.

**SW:** OBS/WebSocket Overlay (Prototype)

* Build WebSocket server for real-time updates (score, eval bar, move suggestions).
* Design lightweight overlay widget (React component or browser source).
* Prototype integration with OBS (as browser source).
* **Deliver initial overlay showing:**
  + Current move number
  + Stockfish top line
  + Eval score

**Deliverable:**

* **AI**: Basic tactics trainer puzzles generated from PGNs.
* **SW**: Overlay prototype working in OBS with test data.

## **Week 7**

**AI: Commentator QuickPack Logic**

* **Generate pre-match packets:**
  + Head-to-head history.
  + Opening frequency & novelty probability.
  + Player tendencies (time usage, blunders).
* Summarize into bullet-point packs for commentators.
* Store/export as JSON/PDF for testing.

**SW: Coach Dashboard (Basic Features)**

* **Build coach UI with:**
  + Roster view (students).
  + Assign lessons/puzzles.
  + Track progress (games analyzed, puzzles solved).
* **Backend**: add tables for coach → student mapping.

**Deliverable:**

* **AI**: QuickPack module generates analyst-ready stats.
* **SW:** Coach dashboard v1 functional.

Milestone: Core architecture + baseline features in place.

## **Week 8**

**Integration Week**

* Connect all Week 1–4 modules (auth, PGN, RL bot baseline, LLM summaries, tactics trainer, QuickPack, overlay, coach dashboard).
* Resolve API contracts between AI & SW.
* Run internal demo:
  + User uploads PGN → bot analysis → tactics generated → summaries visible → overlay works.

Deliverable: Internal MVP Demo.

## **Week 9**

**AI: Adjustable RL Bot Personalities**

* **Implement sliders for:**
  + Aggressive ↔ positional.
  + Blunder-prone ↔ accurate.
  + Fast-moving ↔ slow, time-based profiles.
* Create baseline profiles (e.g., “Club Player,” “Master,” “Random Bot”).

**SW: Frontend Polish + Dashboards**

* **Improve UI/UX:**
  + Responsive design for dashboards.
  + Add charts/graphs for progress tracking.
  + Refine overlay visuals (stream-friendly).

**Deliverable:**

* **AI**: Bot with multiple adjustable personalities.
* **SW**: Polished frontend dashboards with better usability.

## **Week 10**

**AI: Improve Tactics + LLM Summaries**

* Enhance tactics generator with theme classification (pins, forks, mating nets).
* Refine LLM prompts to reduce hallucinations (ground outputs in engine eval).
* Cache summaries for re-usability in commentary packs.

**SW: API Integrations (Lichess, Chess.com)**

* **Connect external APIs for:**
  + Auto-import user games.
  + Fetch player profiles & stats.
* Add OAuth login for seamless onboarding.

**Deliverable:**

* **AI:** More reliable tactics & summaries.
* **SW**: API integrations live.

**Milestone: All features integrated (Phase I).**

## **Week 11**

**AI: Stress Test RL Bots + Caching**

* Run matches with different bot personalities.
* Monitor latency & stability.
* Optimize caching strategy for LLM + engine outputs.

**SW: UI Refinements (Coach/Player Separation)**

* Separate player and coach views.
* Add permissions/roles logic.
* Enhance QuickPack exports (commentator view).

**Deliverable: Stable, role-based UI + optimized AI modules.**

## **Week 12**

**AI + SW:** Batch Testing & Debugging

* Test with bulk PGN datasets.
* **Validate**:  
  + Analysis accuracy.
  + Tactics trainer coverage.
  + QuickPack reliability.
* Debug AI ↔ SW integration issues (data formatting, latency).

**Deliverable: Bugs documented + fixes in progress.**

## **Week 13**

Freeze Development

* No new features → only stabilization & bug fixes.
* Optimize performance (DB queries, caching).
* Harden overlay for low-latency streams.

Deliverable: MVP build frozen (ready for testing).

Milestone: MVP feature complete.

## **Weeks 14–15(Testing & Deployment)**

**Full QA Across User Journeys**

* Player: upload game, train vs bot, review tactics.
* Coach: manage roster, assign, track.
* Commentator: generate QuickPack, use overlay.

**Load/Stress Testing**

* Bulk PGN uploads.
* Concurrent overlay connections.
* RL bot games at scale.

**Bug Fixes + Deployment Prep**

* Final polish, patch critical issues.
* Prepare cloud deployment (Docker, CI/CD).
* Final documentation (user + developer).

## **4. Roles & Responsibilities (RACI Matrix)**

| ***Task/Deliverable*** | ***AI Lead*** | ***AI Eng*** | ***SW Lead*** | ***Frontend Dev*** | ***Backend Dev*** | ***Stakeholders*** |
| --- | --- | --- | --- | --- | --- | --- |
| *RL Bot Development* | ***A/R*** | ***R*** | *C* | *I* | *I* | *I* |
| *LLM Analysis Pipeline* | ***A/R*** | ***R*** | *C* | *I* | *I* | *I* |
| *Tactics Trainer* | ***A/R*** | ***R*** | *C* | *I* | *I* | *I* |
| *Coach Dashboard* | *C* | *I* | ***A/R*** | ***R*** | *R* | *I* |
| *PGN Service & Storage* | *I* | *I* | ***A/R*** | *I* | ***R*** | *I* |
| *OBS Overlay* | *C* | *I* | ***A/R*** | *R* | *R* | *I* |
| *System Integration (Week 5)* | *A* | *R* | ***A/R*** | *R* | *R* | *I* |
| *QA & Testing* | *R* | *R* | *R* | *R* | *R* | *C* |

*(R = Responsible, A = Accountable, C = Consulted, I = Informed)*

## **5. Exit Criteria / Success Metrics**

* **By Week 10:** MVP build feature-complete, stable enough for testing.
* **By Week 12:** Final SaaS deployed with QA sign-off.

**KPIs:**

* Stable release with ≥ 95% uptime in test.
* RL bots playable, LLM analysis accuracy ≥ 80%.
* Coach dashboard handles ≥ 10 players.

## **6. Risks & Mitigation**

* **Risk of time compression** → Strict weekly demos, freeze at Week 10, buffer 2 weeks.
* **Risk of integration delays** → Week 5 and Week 7 integration checkpoints.
* **Risk of testing overruns** → Reserve full 2 weeks solely for QA (no new features).

## **7. Assumptions & Constraints**

### **Assumptions**

* APIs (Chess.com/Lichess) remain accessible.
* Team availability ~35 hrs/week combined.
* Stable compute access for RL + LLM runs.

### **Constraints**

* **Time:** 12 weeks maximum.
* **Budget:** Fixed (compute costs capped).
* **Scope:** Only Phase I features.

## **8. Communication Plan**

* **Reporting Frequency:** Weekly updates + progress log (420 hrs total tracking).
* **Channels:** Slack for daily sync, weekly Zoom review, shared dashboard.
* **Stakeholders:** Informed weekly with report + demo checkpoints (Weeks 5, 8, 12).

## **9. Approval & Sign-Off**

* **Product Owner:** 64 Squares
* **AI Lead / Software Lead:** Sign-off before each milestone.