Here is a detailed 7-day task split for your 4-member development team to deliver the Epic E1: Core Chess Game Play experience, aligned with your technology stack and user stories. This plan ensures daily deliverables for technical lead updates and leverages your preferred stack (Next.js, Tailwind CSS, NestJS, Socket.IO, chess.js + Stockfish, PostgreSQL, Redis, AWS S3, Firebase/Auth0/Passport).

Technology Stack Assessment

Your selected stack is well-suited for this project:

- **Next.js**: Excellent for PWA with SSR and SEO benefits.
- **Tailwind CSS**: Great for custom UI designs and rapid styling.
- **NestJS (Node.js + TypeScript)**: Robust backend framework with modularity and scalability.
- **Socket.IO**: Ideal for real-time chess gameplay communication.
- **chess.js + Stockfish**: chess.js for move validation and game state; Stockfish for AI engine integration.
- **PostgreSQL + Redis**: Reliable for persistent data and caching.
- **AWS S3**: For file storage needs.
- **Firebase Auth/Auth0/Passport**: Flexible authentication options.

No immediate changes are recommended; this stack is modern, scalable, and fits your requirements well.

7-Day Task Breakdown for 4 Developers (D1, D2, D3, D4)

Day 1: Foundation Setup & Core UI Components

- **D1: Project Setup & Chessboard UI**

    - Initialize Next.js project with Tailwind CSS.
    - Implement chessboard UI component with pieces positioned per standard chess rules.
    - Ensure board orientation changes based on player color (white/black perspective).
    - Deliverable: Functional chessboard UI demo.
- **D2: Backend Setup & Basic APIs**
    - Setup NestJS backend project.
    - Design database schema for users, games, moves, and player info.
    - Implement APIs for fetching player info and game state.
    - Deliverable: API endpoints for player info and game initialization.
- **D3: Real-time Communication Setup**
    - Setup Socket.IO server and client integration.
    - Implement basic socket connection and event handling for game start.
    - Deliverable: Real-time connection established with simple message exchange.
- **D4: Player Info Display UI**
    - Implement UI for displaying player username, rating, club affiliation, and captured pieces.
    - Handle guest user display logic (no rating/club).
    - Deliverable: Player info panel integrated with dummy data.

Day 2: Game Clock & Timers + Move Tracking

- **D1: Game Clock & Timer UI**
    - Implement countdown clocks for both players with support for blitz (5 min), rapid (10 min), bullet (3 min).
    - Visual indication when time is running low (color change/blinking).
    - Deliverable: Functional timers with visual urgency indicator.
- **D2: Backend Timer Logic & Synchronization**
    - Implement timer management logic on backend.
    - Sync timers via Socket.IO events to frontend.
    - Deliverable: Backend timer sync with frontend display.
- **D3: Move Tracker UI**
    - Implement move list UI showing all moves in standard algebraic notation.
    - Support scrolling and highlighting current move.
    - Deliverable: Move tracker UI with sample moves.
- **D4: Move Replay Controls**
    - Implement rewind/forward buttons to navigate through past moves.
    - Integrate with chessboard UI to update board state on move navigation.
    - Deliverable: Move replay controls functional.

Day 3: Game Options & Controls

- **D1: Implement Resign, Draw Offer/Accept, Abort**
    - UI buttons for resign, offer/accept draw, and abort game.

    - Backend logic to handle these actions, including acceptance criteria (e.g., abort only before first move).
    - Deliverable: Game options functional with backend integration.
- **D2: Sound Settings Toggle**
    - Implement toggle control for game sound effects.
    - Integrate sound effects for moves, notifications.
    - Deliverable: Sound toggle working with sound effects.
- **D3: Waiting & Pre-Game Screen**
    - Implement waiting screen UI during matchmaking.
    - Include option to cancel matchmaking.
    - Deliverable: Waiting screen with cancel matchmaking.
- **D4: Matchmaking Backend Logic**
    - Implement matchmaking logic for pairing players.
    - Support cancellation of matchmaking.
    - Deliverable: Matchmaking backend with cancel support.

Day 4: In-Game Notifications & Move Validation

- **D1: Turn Indication UI**
    - Visual cues to indicate when it’s the player’s turn.
    - Deliverable: Turn indicator integrated with game UI.
- **D2: Move Validation with chess.js**

- Integrate chess.js to validate moves on frontend and backend.
- Reject illegal moves.
- Deliverable: Move validation logic functional.

- **D3: Real-time Notifications**
  - Implement real-time notifications for move legality, turn changes, and game status updates.
  - Deliverable: Notifications system integrated with Socket.IO.
- **D4: Timer Low-Time Visual Indicator**
  - Enhance timer UI to visually indicate final minute without interfering with gameplay.
  - Deliverable: Timer urgency indicator polished.

Day 5: Game End & Result Display

- **D1: Game Result UI**
  - Display final game result with reason (checkmate, timeout, draw, resignation).
  - Show rating changes for registered users.
  - Deliverable: Game end summary UI.
- **D2: Backend Game End Logic**
  - Implement logic to detect game end conditions (checkmate, timeout, draw rules).
  - Calculate rating changes.

    - Deliverable: Game end backend logic.
- **D3: Handling Disconnections & Aborts**
    - Implement logic for game aborts (pre-first move, disconnection > 2 minutes, server restart).
    - Deliverable: Abort and disconnect handling.
- **D4: Integration Testing Preparation**
    - Prepare test cases for game end scenarios.
    - Deliverable: Test plan and scripts.

Day 6: Integration, Bug Fixes & Polishing

- **All Developers:**
    - Integrate frontend and backend components.
    - Fix bugs found during integration.
    - Improve UI/UX based on initial feedback.
    - Deliverable: Fully integrated and functional core gameplay feature.

Day 7: Buffer & Final Testing

- **All Developers:**
    - Finalize unit, integration, and end-to-end tests.
    - Perform performance and load testing.
    - Prepare deployment scripts and documentation.

    - Deliverable: Tested, stable, and deployable core chess gameplay module.

Summary Table of Daily Tasks per Developer

Day	Developer 1 (D1)	Developer 2 (D2)	Developer 3 (D3)	Developer 4 (D4)
1	Next.js & Chessboard UI	NestJS backend & APIs	Socket.IO setup & basic events	Player info UI
2	Game clocks & timers UI	Backend timer logic & sync	Move tracker UI	Move replay controls
3	Resign/draw/abort UI & backend	Sound toggle & effects	Waiting screen UI	Matchmaking backend
4	Turn indication UI	Move validation (chess.js)	Real-time notifications	Timer low-time visual indicator
5	Game result UI	Game end logic backend	Disconnect & abort handling	Integration test prep
6	Integration & bug fixes	Integration & bug fixes	Integration & bug fixes	Integration & bug fixes
7	Final testing & deployment prep	Final testing & deployment prep	Final testing & deployment prep	Final testing & deployment prep

This plan is designed to keep all team members engaged daily with clear deliverables, enabling steady progress and timely completion of the Core Chess Gameplay epic within 7 days, including 2 days of testing.