Software Requirements Document (SRD)

Project: Tic-Tac-Toe+ (Progressive Web App)

Version: 1.0 (Draft)

Author: Daniel Estrada

Date: October 2025

# 1. Introduction

1.1 Purpose  
This document defines the requirements for Tic-Tac-Toe+, a Progressive Web App (PWA) game designed for 1–2 players on mobile and desktop. It describes the scope, functional and non-functional requirements, design goals, and acceptance criteria.

1.2 Scope  
Tic-Tac-Toe+ is a modernized version of the classic Tic-Tac-Toe game with new modes, power-ups, and a polished user interface. It will run entirely client-side, be installable as a PWA, and work offline.  
  
The project will be open source and hosted on GitHub with GitHub Pages deployment.

1.3 Goals  
- Deliver a lightweight, fun, installable Tic-Tac-Toe experience.  
- Offer Classic (3×3), Plus (4×4), and Power Mode (with power-ups).  
- Support 1 player (AI) and 2 players (local turns).  
- Provide persistent stats tracking.  
- Achieve high Lighthouse scores (≥90 for PWA, Performance, Accessibility).

# 2. System Overview

- Platform: Web (installable PWA, offline-first).

- Framework: React + Vite + TypeScript (recommended).

- Hosting: GitHub Pages.

- Architecture: Modular front-end only, no backend services.

- Data Storage: LocalStorage or IndexedDB.

# 3. Functional Requirements

3.1 Game Modes  
- Classic Mode (3×3): Standard rules, win with 3 in a row.  
- Plus Mode (4×4): Win with 3 in a row on a larger grid.  
- Power Mode (4×4): Includes limited-use power-ups.

3.2 Power-Ups (Power Mode Only)  
- Swap: Convert one opponent piece to your own.  
- Bomb: Remove one opponent piece.  
- Double-Move: Place two marks in one turn.  
(Each can be used once per player per match.)

3.3 Players & Turns  
- 1 Player vs AI: AI difficulty levels: Casual (random + heuristics), Smart (minimax for 3×3).  
- 2 Player Local: Alternating turns on the same device.  
- Timer Option: 5–10 second turn timer.

3.4 Match Management  
- Best-of-1, Best-of-3, or Best-of-5.  
- Undo (Classic mode only, one per game).  
- Draw detection.

3.5 Stats  
- Persistent win/loss/draw counters.  
- Streak tracking.  
- Reset option in settings.

3.6 User Interface  
- Home screen with Play, Modes, Stats, Settings, Help.  
- In-game HUD with board, turn indicator, timer, and power-up tray.  
- Results popup (Win/Draw).  
- Animated win-line highlights.

3.7 Accessibility  
- Screen reader support (announcements for moves, wins).  
- Keyboard navigation.  
- High-contrast theme option.  
- Reduced motion option.

# 4. Non-Functional Requirements

- Performance: Load in under 2s on first visit; under 1s on repeat.

- Offline: Game fully playable offline.

- Accessibility: WCAG AA compliance.

- Security: No external data collection.

# 5. System Design

5.1 Architecture  
- Game Core Module: Rules engine, win/draw detection, power-up logic.  
- AI Module: Minimax (3×3), heuristic AI (4×4).  
- UI Module: React components for board, HUD, dialogs.  
- State Management: Zustand/Context API.  
- PWA Module: Service worker, manifest, icons.

5.2 Data Model (simplified)  
```  
type Player = 'P1' | 'P2' | 'AI';  
type Cell = 'X' | 'O' | null;  
type Board = Cell[][];  
  
interface GameState {  
 mode: 'classic3' | 'plus4' | 'power4';  
 board: Board;  
 current: Player;  
 powerups: { swap: boolean; bomb: boolean; double: boolean };  
 status: 'playing' | 'win' | 'draw';  
 winner?: Player;  
}  
```

5.3 Storage  
- LocalStorage keys:  
 - settings (theme, AI level, timer)  
 - stats (per mode win/loss/draw, streaks)

# 6. PWA Requirements

- Installable via manifest + service worker.

- Offline-first (app shell cached).

- Icons (192x192, 512x512, maskable).

- “New version available” refresh prompt.

# 7. Testing

- Unit Tests: Game logic, win conditions, power-up effects.

- Integration Tests: Board interactions, AI moves.

- E2E Tests: Start → play → win/draw; offline mode.

- Accessibility Tests: Keyboard, screen reader, contrast.

- Performance Tests: Lighthouse automation in CI.

# 8. Deployment

- CI/CD: GitHub Actions build, test, deploy pipeline.

- Hosting: GitHub Pages (gh-pages branch).

- Versioning: Semantic Versioning (SemVer).

# 9. Milestones

- M1: Classic mode (3×3), local play, AI, PWA basics.

- M2: Plus mode (4×4), stats, themes.

- M3: Power mode, power-ups, polish, release v1.0.

# 10. Risks

- AI performance on 4×4 → keep simple heuristic for MVP.

- Service worker caching bugs → add refresh prompt.

- Accessibility gaps → test with screen readers before release.

# 11. Acceptance Criteria

- Game runs offline, installable on mobile.

- Classic, Plus, and Power modes functional.

- Stats persist after reload.

- At least one AI difficulty works as expected.

- Passes Lighthouse audit with ≥90 in all categories.

# 12. Appendices

12.1 Manifest Example  
```  
{  
 "name": "TicTacToe+",  
 "short\_name": "T3+",  
 "start\_url": "/",  
 "display": "standalone",  
 "background\_color": "#0b0f19",  
 "theme\_color": "#0ea5e9",  
 "icons": [  
 { "src": "/icons/icon-192.png", "sizes": "192x192", "type": "image/png" },  
 { "src": "/icons/icon-512.png", "sizes": "512x512", "type": "image/png" },  
 { "src": "/icons/maskable-512.png", "sizes": "512x512", "type": "image/png", "purpose": "maskable" }  
 ]  
}  
```