PRD — Voxel Worlds (Web Minecraft-Style Builder)

MVP for Chrome (desktop) + Chrome (iPadOS) • 3 starter world themes • <=5 players per room

Version: v5 • Updated Feb 6, 2026 (America/La_Paz)

Owner	CTO (you)
Stakeholders	Product, Engineering, Design, QA
Target Release	MVP
Last Updated	February 6, 2026

Executive Summary

Voxel Worlds is a lightweight, browser-based voxel builder inspired by Minecraft. Users sign up, create a private room, set a world password, share an invite link, and collaboratively build while talking over proximity voice chat with spatial audio. The MVP prioritizes performance and simplicity: deterministic terrain generation (seeded), place/break blocks, a small curated block palette, multiplayer sync for <=5 players, and world saving.

MVP Decisions (Locked)

- Auth + lobby: Next.js + Supabase (Auth via email OTP, Postgres, Storage, Realtime signaling/presence).
- Realtime high-frequency sync: WebRTC DataChannels (mesh topology for <=5). Supabase Realtime is used for signaling and low-frequency fallback.
- Voice: WebRTC audio; spatialization via Web Audio API (PannerNode + GainNode).
- Audio default: open mic with self-mute toggle (push-to-talk out of MVP).
- World state: base terrain generated locally from (theme, seed); only diffs (block edits + snapshots) are synced/saved.
- Rooms are persistent. Each account can own up to 3 worlds total (create/delete to manage the cap).
- Host-only saves + moderation: host can save/resume and can kick guests from the room.
- WebRTC reliability: TURN is enabled and required for voice + data connectivity (coturn or managed).

1. Overview

Voxel Worlds is a small-session voxel builder for private groups. Users can: sign up, create a room, set a world password, share an invite, join from Chrome (desktop) or Chrome (iPadOS), build together, and talk with spatial voice. The MVP does not include survival, crafting, mobs, or complex mechanics.

Starter world themes (MVP):

- Forest (Canadian style) — dense conifers, lakes, mossy rocks, warm lighting
- Snow (Norway style) — snowy terrain, fjord-like valleys, pine forests, aurora skybox
- Coast (Greece style) — sandy/rocky shorelines, white-stone props, blue water, Mediterranean palette

2. Goals and Non-Goals

2.1 Goals (MVP)

- Signup/login + dashboard to create/manage up to 3 persistent worlds and join invited worlds.
- Room creation: world theme, seed (auto), world password, invite link; max players fixed to 5 for MVP.
- Multiplayer building: see other players, place/break blocks, synced world diffs.
- In-session voice chat with spatial audio based on avatar distance and direction.
- Cross-device: Chrome (desktop) and Chrome (iPadOS) with touch controls.
- Performance-first rendering with chunking and view distance suitable for web.
- Persist worlds (save and resume later) with minimal storage cost.

2.2 Non-Goals (MVP)

- Survival gameplay: hunger/health, crafting, tools, combat, mobs.
- Large-scale multiplayer (>5 players) or public matchmaking.
- Modding, custom texture packs, marketplace.
- Advanced physics (fluids, circuits, complex lighting) or redstone-like systems.
- Strong anti-cheat beyond basic validation and abuse protections (private sessions only).

3. Target Users and Use Cases

3.1 Personas

- Creator Host: creates rooms, invites friends/colleagues, leads a build session.
- Guest Builder: joins via invite link + password, collaborates on a shared build.
- Casual Team: small group using the world as a playful collaboration space.

3.2 Primary Use Cases

1. User signs up -> creates a Forest world -> shares invite link + password -> friends join -> build together while talking.

2. User joins a Snow world from Chrome (iPadOS) -> uses touch controls to build -> voice chat works with spatial audio.

3. Host resumes a saved Coast world -> continues building with a new group.

4. Success Metrics

- Activation: % of signed-up users who create or join a room within 10 minutes.
- Session reliability: % of sessions where all players remain connected >=10 minutes without voice failure.
- Performance: median FPS >=50 desktop, >=30 Chrome (iPadOS); join-to-play median <20s on broadband.
- Collaboration: average block edits per user per session (proxy for engagement).
- Retention: % returning to resume a saved world within 7 days.

5. MVP Scope

5.1 World Themes (3)

Each theme ships with a deterministic generator configuration, texture palette, skybox, and ambience. Given a seed, every client must generate identical base terrain.

- Forest: rolling hills, lakes/streams, conifer trees, rocks; mild fog; warm sunlight.
- Snow: snow layer, ice patches, fjord valleys, pine trees; cold lighting; optional aurora skybox.
- Coast: beaches, cliffs, shallow water, sparse trees; bright sunlight; coastal haze; white-stone props.

5.2 Construction (Core Gameplay)

- Creative-mode building with a curated palette (10-20 block types; no stairs/slabs/decoratives in MVP).
- Block place and break (single block).
- Hotbar (1–9) and basic inventory picker panel.
- Basic collision: player cannot walk through solid blocks.
- Simple gravity + jump.

5.3 Multiplayer

- Up to 5 concurrent players per room.
- Player presence: names, simple avatar (capsule), position + orientation.
- World diffs: block changes synchronized near real-time.

- Late joiners receive current world state (base seed + diffs).

5.4 Voice + Spatial Audio

- Voice enabled in session (open mic by default + self-mute toggle).
- Proximity: within radius (default 25m) voice attenuates with distance; beyond radius becomes silent.
- Graceful fallback when mic permission denied (user can still play).

5.5 Devices and Input

- Desktop: keyboard/mouse with pointer lock.
- iPad (Chrome iPadOS): touch controls (on-screen joystick, swipe look, place/break buttons).
- Basic accessibility: rebind controls on desktop; captions out of scope for MVP.

6. Out of Scope (Explicit)

- Public discovery/browse worlds.
- In-world economy, monetization mechanics, skins shop.
- Streaming/recording integrations.
- Dedicated phone support (best effort only).

7. User Flows

7.1 Signup/Login

1. Landing -> Sign up -> verify (if needed) -> dashboard.

2. Returning user -> login -> dashboard.

7.2 Create Room (Host)

1. Dashboard -> Create Room -> select theme -> optional custom seed -> set world password -> Create.

2. Room created -> show share link + copy -> Launch -> game loads.

7.3 Join Room (Guest)

1. Open invite link -> enter world password -> join -> lobby -> Launch -> game loads.

7.4 In-Game Loop

1. Spawn -> move/look -> select block -> place/break -> see others update.

2. Voice proximity changes as players move.

3. Pause -> settings -> leave (host triggers save).

8. Functional Requirements

8.1 Landing Page (Next.js)

- Responsive landing page: hero, features, media placeholders, FAQ, footer (terms/privacy).
- SEO basics: meta tags, open graph tags, sitemap, robots.

8.2 Auth & Account

- Supabase Auth using email OTP (one-time code sent to email; no password in MVP).
- User profile: display name required; avatar color required (auto-assigned; user can change). Name tag is shown above the avatar.
- Account settings: change display name; sign out; delete account (soft delete acceptable).

8.3 Room Management

- Create room: name, theme, seed, max players (fixed=5), invite token, password hash.
- Invite link contains roomId + invite token (unguessable).
- World password required in addition to invite token.
- Host controls: end session, save world, rotate password (optional in MVP).
- World cap: each account can own up to 3 worlds total; UI enforces and backend validates the limit.
- Moderation: host can kick a guest (disconnect them from voice/data and remove from presence).

Security note: world password must never be stored in plaintext. Store a salted hash and validate server-side (Edge Function or server route).

8.4 World Persistence

- Store base world: theme + seed.
- Store diffs: chunk-level delta snapshots OR event log compacted periodically.
- Save-on-exit by host and resume later.
- World file size target: <10MB per typical 30-minute build session (compressed).

8.5 Multiplayer Sync

- Deterministic base generation; transmit only player state + block edit events.
- Preferred: WebRTC DataChannels for high-frequency updates; fallback to WebSocket (Supabase Realtime or custom) if WebRTC fails.
- Supabase Realtime used mainly for signaling, lobby presence, and low-frequency fallback events.
- Network messages versioned and validated; reject malformed packets.

8.6 Voice Chat (WebRTC)

- WebRTC audio streams between participants (mesh topology for <=5).
- Signaling per room (offer/answer/ICE).

- TURN is required and enabled for WebRTC reliability (self-hosted coturn or managed provider).

8.7 Spatial Audio (Web Audio API)

- Remote audio track -> MediaStreamAudioSourceNode -> PannerNode (HRTF) -> GainNode -> output.
- Update panner positions from avatars at 10–20 Hz with smoothing.
- Expose settings: master voice volume, proximity radius (optional in MVP), mute self.

9. Non-Functional Requirements

9.1 Performance Targets

- Desktop: 50–60 FPS typical; <200ms input-to-render for block placement.
- Chrome (iPadOS): 30+ FPS typical; conservative defaults (lower view distance).
- Join time: invite click to spawn <=20 seconds on broadband.
- Networking: graceful degradation at 200–300ms RTT (interpolation).

9.2 Rendering/Engine Constraints

- Chunked voxels (e.g., 16x16x32 or 16x16x64), frustum culling, configurable view distance.
- Greedy meshing (or similar) to reduce triangles; avoid one-cube-per-face rendering.
- Web Workers for meshing and noise computations.
- Texture atlas to reduce draw calls; limit dynamic lights; quality toggles.

9.3 Security & Privacy

- Password hashing and server-side validation only.
- Supabase RLS: only room members can read room details/members/world metadata.
- Invite tokens random UUIDs; prevent room ID enumeration.
- Rate-limit join attempts to reduce password brute force.
- No voice recording; no voice data stored.

10. Technical Architecture (Proposed)

10.1 Components

- Next.js: marketing + app UI + server routes for password validation and optional signaling.
- Three.js client: voxel engine, HUD, networking, audio.
- Supabase: Auth, Postgres DB, Storage for saves, Realtime signaling/presence.
- TURN (coturn) is required for WebRTC reliability.

10.2 Data Flow (Happy Path)

1. User authenticates via Supabase Auth.

2. Host creates room -> DB row created -> invite token generated.

3. Guests open invite -> password validated server-side -> membership created.

4. Clients connect to signaling channel -> establish WebRTC connections (audio + data).

5. Clients generate base terrain from (theme, seed); host sends latest snapshot + remaining events to late joiners.

6. Block edits broadcast over data channel; periodic snapshots saved to Supabase Storage.

10.3 Supabase Data Model (MVP Tables)

Table	Key Columns	Purpose	RLS Notes
profiles	user_id (PK), display_name, avatar_color	User profile metadata	Owner read/write; limited read to room members
rooms	id (PK), host_id, theme, seed, status, invite_token, password_hash, created_at	Room metadata	Readable only to members + host
room_members	room_id, user_id, role, joined_at	Membership list and roles	Members can read; host can manage
world_saves	room_id, save_id, storage_path, created_at, created_by	Save references	Members can read; host can write
block_events (optional)	room_id, seq, chunk_id, payload, created_at	Append-only edit log	Members can read/write (rate limited)

10.4 Realtime Channels

- presence:room:{roomId} — lobby presence list.
- signal:room:{roomId} — WebRTC signaling (offer/answer/ICE).
- events:room:{roomId} — low-frequency fallback events (if WebRTC data is unavailable).

11. Multiplayer & Sync Details

11.1 World State

- Base: deterministic from (theme, seed).
- Edits: events (place/break) with coordinates, block type, actor, and timestamp.
- Compaction: host generates chunk-diff snapshots every N events or T minutes.
- Late join: base -> latest snapshot -> replay remaining events.

11.2 Player State

- Send position/orientation at 10–20 Hz.
- Interpolate remote avatars; dead reckoning for smoother motion.
- Send discrete actions (place, break, jump) as events for correctness.

11.3 Conflict Handling

- Host assigns monotonically increasing sequence numbers for block edits.
- If two edits hit the same block concurrently: last-write-wins by sequence.
- If host disconnects: allow reconnect within grace period; otherwise end session (MVP).

12. Voice + Spatial Audio Details

12.1 Topology

- Mesh WebRTC for <=5: each client maintains up to 4 peer connections.
- Audio only (no video) to reduce bandwidth and CPU.

12.2 Spatialization

- Attenuation radius default 25m; future: per-room setting.
- Directional cues via HRTF panning; listener orientation follows camera.
- Occlusion is out of scope for MVP.

12.3 UX

- Request microphone permission on world entry with clear explanation.
- If denied: user can still play; show muted state and allow retry.
- Default: open mic with echo cancellation/noise suppression + self-mute toggle (push-to-talk out of MVP).

13. UX/UI Requirements (MVP)

13.1 Pages

- Landing (/): hero, features, media placeholders, FAQ, footer.
- Auth (/signup, /login).
- Dashboard (/app): owned worlds list (max 3), create world, join via invite.
- Room detail (/app/rooms/{id}): share link, members, launch.
- Game (/play/{roomId}): full-screen canvas + HUD overlay.

13.2 In-Game HUD

- Crosshair, hotbar, selected block label.
- Voice status: mic on/off, speaking indicator (optional), per-user mute (optional).
- Menu: settings, controls help, leave session.

13.3 iPad Controls (Chrome iPadOS)

- Left joystick move; right swipe look.
- Buttons: jump, place, break, inventory/hotbar toggle.
- Touch targets sized for tablets; avoid hover-only interactions.

14. Analytics & Logging (MVP)

- Events: signup, room_created, room_joined, session_started, voice_enabled, blocks_placed, blocks_broken.
- Error logging: WebRTC failures, asset load failures, unhandled exceptions.
- Privacy: avoid collecting unnecessary identifiers; document in privacy policy.

15. Testing Strategy

15.1 Automated

- Unit: world gen determinism, meshing, edit application, message validation.
- Integration: Supabase RLS policies, room creation/join/password validation.
- E2E (Playwright): signup -> create room -> join from second context -> edit block -> verify sync.

15.2 Manual QA

- Chrome (desktop): pointer lock, movement, edit loop, HUD, voice + proximity effect.
- Chrome (iPadOS): controls usable, layout stable, voice permission prompts, acceptable FPS.
- Networking: reconnect after Wi-Fi drop; late join; host save + resume.

16. Risks & Mitigations

- WebRTC reliability on restrictive NATs/mobile networks: require TURN + reconnection logic.
- Supabase Realtime not ideal for high-frequency sync: use WebRTC DataChannels for player state/edits; Supabase only for signaling and fallback.
- Voxel performance on iPad: greedy meshing, conservative view distance, quality toggles, minimize draw calls.
- World save bloat: snapshots + compression + periodic compaction.

17. Delivery Plan (Epics, Tasks, DoD)

The following epics are structured for delegation to dev agents. Each epic includes a Definition of Done.

Epic E1 — Product Foundation (Next.js + Supabase)

Goal: Users can sign up, log in, and access the dashboard.

- E1.1 App scaffold (Next.js App Router), linting/formatting, env configuration.
- E1.2 Supabase Auth integration (email OTP signup/login/logout).
- E1.3 Profiles table + RLS + UI to set display name.
- E1.4 Landing page + Terms/Privacy placeholders.

Definition of Done:

- Signup/login works end-to-end and routes correctly.
- RLS policies verified (tests or manual SQL validation).
- README documents local setup and required env vars.

Epic E2 — Rooms (Create/Join/Invite/Password)

Goal: Host creates a persistent world (<=3 worlds per account); guests join with invite + password.

- E2.1 rooms + room_members schema + RLS.
- E2.2 Create Room UI (theme selection, password).
- E2.3 Invite link generation + copy-to-clipboard.
- E2.4 Password validation endpoint (Edge Function or Next.js server route) using password_hash.
- E2.5 Room detail page with member list + launch button.
- E2.6 Worlds dashboard: list owned worlds (max 3), launch/resume, delete world (soft delete acceptable).
- E2.7 Host kick control (lobby + in-game) with UI + backend enforcement.

Definition of Done:

- Guest cannot join without correct password even with invite.
- Room member list is accurate; host role enforced.
- Basic abuse protection: join attempts rate-limited.
- World cap enforced: a user cannot create a 4th world (UI + backend).
- Host can kick a guest and the guest is removed from presence and disconnected from voice/data until they rejoin.

Epic E3 — Voxel Engine Core (Three.js)

Goal: Single-player world loads, player moves, and blocks can be placed/broken.

- E3.1 Three.js scene setup, camera, lighting, skybox; robust resize.
- E3.2 Chunk system + deterministic world generation (seeded noise).
- E3.3 Greedy meshing + Web Worker pipeline.
- E3.4 Player controller (desktop + iPad touch controls for Chrome iPadOS).
- E3.5 Block interaction (raycast, place/break) + hotbar/inventory HUD.

Definition of Done:

- Forest world loads deterministically for the same seed.
- Place/break updates the mesh instantly without corruption.
- Desktop median >=50 FPS with default settings; iPad quality mode available.

Epic E4 — Multiplayer Sync (<=5)

Goal: Two clients join the same room and see each other + synced edits.

- E4.1 Supabase Realtime signaling channel per room.
- E4.2 WebRTC peer connections (audio + data).
- E4.3 Player state replication at 10–20 Hz with interpolation.
- E4.4 Block edit events with host-ordered sequencing.
- E4.5 Late join snapshot + event replay.

Definition of Done:

- Two clients on different networks can connect using TURN (required/enabled).
- Edits replicate within <500ms typical on WAN.
- Late joiner reaches correct world state within 10 seconds.

Epic E5 — Voice + Spatial Audio

Goal: Voice works and audibly changes with distance.

- E5.1 Mic capture with echo cancellation + noise suppression.
- E5.2 Remote tracks routed through spatial pipeline (PannerNode + GainNode).
- E5.3 Audio positions updated from avatar positions with smoothing.
- E5.4 HUD controls: mute/unmute + volume.

Definition of Done:

- Voice works on Chrome (desktop) and Chrome (iPadOS) in at least one realistic NAT scenario with TURN enabled.
- Spatial effect is obvious in a simple two-player test.
- Mic denied does not prevent gameplay; UI indicates muted state.

Epic E6 — World Saves (Persist & Resume)

Goal: Host saves and resumes the same world later.

- E6.1 Snapshot format (chunk diffs) + compression (gzip/brotli).
- E6.2 Upload to Supabase Storage; references stored in world_saves.
- E6.3 Load latest save on launch; schema versioning.

Definition of Done:

- World persists across sessions and reloads accurately.
- Save files remain under target size for typical sessions.

- Backward compatibility within MVP versions is documented.

Epic E7 — Hardening, Monitoring, Release

Goal: Ship a stable MVP with docs and baseline monitoring.

- E7.1 Error tracking (Sentry or equivalent) + actionable logs.
- E7.2 Quality toggles (view distance, shadows) + defaults per device.
- E7.3 Playwright E2E happy-path suite in CI.
- E7.4 Deployment pipeline + README for local dev and production.

Definition of Done:

- New developer can run locally in <15 minutes following README.
- CI green on main branch; smoke tests validated.
- Known limitations and future work tracked as issues.

18. Decisions Locked and Remaining Questions

Locked decisions (confirmed):

- Auth: email OTP via Supabase Auth (one-time code).
- Room lifecycle: persistent worlds (reusable).
- World limit: each account can own up to 3 worlds total.
- Saves: host-only (save/resume controlled by host).
- Connectivity: TURN required/enabled for WebRTC voice + data.
- Device target: Chrome (desktop) and Chrome (iPadOS) only for MVP.
- Moderation: host can kick guests (ban is out of MVP).
- Audio: open mic default with self-mute toggle (push-to-talk out of MVP).
- Blocks: curated 10-20 block palette only (no stairs/slabs/decoratives in MVP).
- Identity: capsule avatar with user-selected color + name tag.

Remaining questions: none (MVP scope locked).

19. Appendix

19.1 Suggested Block Types (MVP)

- Grass, Dirt, Stone, Sand, Water, Wood Log, Planks, Leaves, Snow, Ice, White Stone/Clay, Cobblestone.

19.2 Quality Presets

- Low (iPad default): view distance 4 chunks, shadows off.
- Medium: view distance 6 chunks, minimal shadows.
- High (desktop): view distance 8 chunks, optional shadows.