Story Clash: Codex Build Specification

Multiplayer Story Battle Game – App Store #1 Vision

Executive Vision

Story Clash is a real-time multiplayer narrative game where 3-6 players compete to survive an intense, branching story. Think "Jackbox meets Choose Your Own Adventure meets Battle Royale storytelling." The experience is fast, visceral, and social—designed for friend groups, streamers, and casual gamers who want a 10-15 minute adrenaline-packed session they'll replay weekly.

Core hook: A quick skill-based minigame determines who controls the story fate and what genre nightmare you're all thrown into. Then players take turns making life-or-death choices under time pressure while others watch, react, and see consequences unfold in real time.

Monetization: Free base game with 3 starter genres. Premium genre packs ($2.99 each or $9.99 bundle), cosmetic avatar packs, and optional "Story Vault" subscription ($4.99/mo) for exclusive weekly story drops and early access.

Target: iOS App Store featured placement. Optimized for viral social sharing, Twitch/YouTube content creation, and high replay value.

Product Requirements (V1 Launch Scope)

1. Onboarding & Lobby System

Goal: Get 3-6 friends into a room in under 30 seconds.

Screens

1.1 Home Screen

- Bold, dark theme with neon accent colors (electric blue, blood red, toxic green).
- Large hero text: "Story Clash" with animated glitch effect.
- Two giant buttons:
  - "Create Room" (primary CTA, glowing border animation).
  - "Join Room" (secondary, subtle pulse).
- Bottom: "How to Play" link (opens modal with 60-second animated tutorial).
- Background: abstract animated particles suggesting chaos/narrative threads.

1.2 Create Room Flow

- Player enters display name (max 12 chars, profanity filter).
- Generates 4-character room code (uppercase letters, no confusing chars like O/0).
- Shows lobby screen immediately.

1.3 Join Room Flow

- Input field for 4-char code, auto-uppercase, validates in real-time.
- Player enters display name.
- On valid code, joins lobby instantly.

1.4 Lobby Screen

- Header: Room code (large, copyable with one tap).
- Player list:
  - Shows all joined players with avatars (simple geometric shapes + colors for V1).
  - Host has crown icon.
  - Each player slot shows "Waiting..." for empty slots.
  - Max 6 players, min 3 to start.
- Big "Share Room" button (generates shareable link, copies code to clipboard).
- Host controls: "Start Game" button (only enabled when 3+ players, glows when ready).
- Ambient sound: low tension drone, subtle heartbeat rhythm.

Technical Requirements:

- Real-time sync via WebSockets (use Socket.io or similar).
- Room expires after 30 minutes of inactivity.
- Reconnection logic: if player drops, they can rejoin within 2 minutes using same name + code.

---

2. Minigame: "Reflex Roulette"

Goal: 20-second intense skill challenge that determines turn order and story genre. Must feel fair, exciting, and watchable.

Mechanics

Concept: Reaction-time challenge with escalating difficulty.

Flow:

1. 3-2-1 countdown (full screen, bold numbers, sound cue each second).
2. A circular target appears center screen with a rotating needle (like a speedometer).
3. A "hit zone" (colored arc) rotates around the circle.
4. Players tap/click when the needle enters the hit zone.
5. Three rounds, each faster.
6. Score = accuracy × speed bonus.

Scoring:

- Perfect hit (dead center of zone): 100 pts + speed multiplier.
- Good hit (edges): 50-80 pts.
- Miss: 0 pts, needle resets.
- Final score = sum of 3 rounds.

Results Screen:

- Leaderboard animates in, sorted by score.

- Top scorer gets "Story Master" badge.
- Turn order = leaderboard rank.
- Genre selection:
  - Top scorer sees 3 genre cards (face-down, dramatic reveal).
  - Taps one to reveal.
  - Genre is announced with full-screen title card + thematic sound effect.

Genres (V1):

1. **Zombie Outbreak** (survival horror, gritty, desperate).
2. **Alien Invasion** (sci-fi horror, paranoia, technology).
3. **Haunted Manor** (gothic horror, mystery, psychological).

Technical Requirements:

- 60fps animation on target/needle.
- Input lag <50ms.
- Mobile-optimized touch zones (large tap area).
- Sound: rising tension music during minigame, dramatic sting on each tap, triumphant/ominous reveal music for genre.

---

3. Story Phase: Turn-Based Narrative

Goal: Create edge-of-your-seat tension as players make choices that shape a branching story under time pressure.

Core Loop

3.1 Scene Display

- Full-screen story text (2-4 sentences, punchy, present-tense, second-person: "You hear glass shatter upstairs. The power cuts out.").
- Animated typewriter effect (fast but readable, ~30 chars/sec).
- Background: dynamic abstract visual tied to genre (flickering shadows for horror, static/glitch for sci-fi).
- Ambient sound: genre-specific soundscape (distant groans, radio static, wind, etc.).

3.2 Active Player Turn

- Turn indicator: "Your Turn, [Name]" banner at top.
- 30-second countdown timer (circular, shrinks with warning colors: green → yellow → red in last 10 sec).
- Player sees:
  - **Two preset choice buttons** (large, high contrast):
    - Example: "Search the basement" vs "Barricade the door."
    - Hover/tap shows subtle preview hint (not outcomes, just tone: "Risky" vs "Safe").
  - **One "Free Choice" option** (text input, placeholder: "Or describe your own action...").
    - Max 60 characters.
    - Submit button appears when text entered.
- If timer expires: random preset choice auto-selected.

3.3 Other Players (Spectating)

- See same scene text.
- See active player's name + timer.
- Can't interact, but UI shows "Waiting for [Name]..." with animated ellipsis.
- Optional: simple emoji reactions (V2 feature, not V1).

3.4 Choice Resolution

- Choice submitted → brief suspense beat (1-2 seconds, screen darkens, sound cue).
- Next scene generated and displayed to all players.
- Turn advances to next player in order.

3.5 Story Generation Logic

- Use predefined branching story trees for V1 (not fully AI-generated to control quality and pacing).
- Each genre has 3-5 branching paths with 8-12 total scenes.
- Each preset choice maps to a specific next scene.
- Free choice: map player input to closest matching branch using keyword detection (e.g., "hide" → stealth path, "fight" → aggressive path).
- Story escalates in intensity: early choices are exploratory, mid-game introduces danger, final act is climax/resolution.

Technical Requirements:

- Story tree stored as JSON (nodes with scene text, choices, next node IDs).
- Timer synced across all clients (server-authoritative).
- Choice submission locks immediately to prevent double-input.

---

4. Ending & Recap

Goal: Deliver satisfying closure and encourage replay/sharing.

4.1 Story Conclusion

- Final scene plays out based on cumulative choices.
- Three possible endings per genre:
  - **Triumph:** Group survives/wins (uplifting music, bright visuals).
  - **Survival:** Barely made it (tense relief, flickering light visuals).
  - **Doom:** Everyone dies/fails (ominous music, fade to black).

4.2 Recap Screen

- Full story playback:
  - Scrollable timeline showing each scene + player who chose + their choice.
  - Highlighted key turning points.
- MVP (Most Valuable Player): player who made the "best" choice (predefined per story path, e.g., choice that led to survival).
- Big "Play Again" button (returns to lobby with same players).
- "Share Story" button (generates shareable image: story title, player names, ending type, app download link).

Technical Requirements:

- Recap data stored in session (scenes, choices, players).
- Share generates Open Graph-compliant image (1200×630px) server-side or client-side canvas render.

---

5. UI/UX Design System

App Store #1 Aesthetic = **Premium** + **Accessible** + **Viral**

Visual Identity

Color Palette:

- **Primary Background:** Deep charcoal (#1a1a1d).
- **Surface/Cards:** Soft black (#26262a) with subtle inner shadow.
- **Text Primary:** Off-white (#f0f0f0).
- **Text Secondary:** Mid-gray (#a0a0a5).
- **Accent 1 (Primary CTA):** Electric cyan (#00d9ff), glows on hover.
- **Accent 2 (Danger/Urgency):** Blood red (#ff3b3b).
- **Accent 3 (Success):** Neon green (#39ff14).

Typography:

- **Headers:** Bold sans-serif, tight letter-spacing, all-caps for emphasis (e.g., "Poppins Bold").
- **Body/Story Text:** Readable sans-serif, slightly larger than default (16-18px base, e.g., "Inter").
- **UI Labels:** Medium weight, clear hierarchy.

Animations:

- Smooth, snappy (200-300ms ease-out for transitions).
- Micro-interactions: buttons scale slightly on tap, glowing pulses on CTAs.
- Story text: typewriter effect, scene transitions fade with subtle zoom.
- No long load screens: use skeleton loaders or instant transitions.

Sound Design:

- **Lobby:** Ambient drone, low-key tension (think Stranger Things menu).
- **Minigame:** Rising synth, heartbeat rhythm, climactic sting on completion.
- **Story Phase:** Genre-specific soundscapes:
  - Zombie: distant moans, creaking wood, wind.
  - Alien: radio static, mechanical hums, distant sirens.
  - Haunted: whispers, footsteps, old house creaks.
- **UI Sounds:** Subtle clicks, swooshes, dings (never annoying, always satisfying).
- **Mute toggle:** Always accessible, settings icon in corner.

Accessibility:

- Font size: minimum 16px, scalable.
- High contrast mode option (for colorblind users).
- Haptic feedback on key actions (iOS).
- Screen reader support for all text and buttons.

---

6. Technical Stack & Architecture

Platform: Web-first (PWA), optimized for iOS Safari + Chrome, then native iOS app via Capacitor or React Native.

Frontend:

- **Framework:** Next.js 14+ (React, App Router).
- **Styling:** Tailwind CSS + Framer Motion for animations.
- **State Management:** Zustand or Jotai (lightweight, fast).
- **Real-time:** Socket.io client.

Backend:

- **Framework:** Node.js + Express or Next.js API routes.
- **Real-time:** Socket.io server.
- **Database:** PostgreSQL (via Supabase or Railway) for:
    - User accounts (optional for V1, required for V2 profiles/stats).
    - Story trees (JSON storage).
    - Session logs (for analytics).
- **Hosting:** Vercel (frontend + serverless functions) + Railway/Render (WebSocket server).

Story Data:

- Store story trees as JSON files or database records.
- Structure:
  ```
  {
  "genre": "zombie_outbreak",
  "scenes": [
  {
  "id": "scene_01",
  "text": "You wake up in a dark room. The air smells of decay.",
  "choices": [
  {"text": "Search for a weapon", "next": "scene_02a"},
  {"text": "Try to find an exit", "next": "scene_02b"}
  ],
  "free_choice_keywords": {
  "weapon|arm|fight": "scene_02a",
  "exit|leave|run": "scene_02b"
  }
  }
  ]
  }
  ```

Performance Targets:

- Initial load: <2 seconds on 4G.
- Real-time latency: <100ms for choice submission → next scene.
- 60fps animations on mid-tier mobile devices.

7. App Store Optimization (ASO) Strategy

Goal: Hit #1 in Games > Word or Games > Puzzle category, featured on App Store.

App Store Listing

App Name: Story Clash: Multiplayer Tales

Subtitle: Survive Intense Story Battles

Keywords: multiplayer story game, choose your own adventure, narrative game, party game, story battle, interactive fiction, group game, horror stories, mystery game, storytelling app

Description (First 170 chars - appears without "more"):
"Compete in intense multiplayer story battles! Play a minigame, make life-or-death choices, and survive branching narratives with friends. Every choice matters."

Full Description:
Story Clash is the ultimate multiplayer narrative experience. Gather 3-6 friends, compete in a lightning-fast minigame to determine who controls fate, then dive into an intense branching story where every choice could be your last.

FEATURES:
• Fast 10-15 minute sessions perfect for game night
• Competitive minigame determines turn order and story genre
• Choose from preset options or create your own path
• Real-time multiplayer with friends anywhere
• Immersive sound design and atmospheric visuals
• Multiple story genres: horror, sci-fi, mystery, and more
• No ads, no pay-to-win—pure storytelling competition

Perfect for:
✓ Friend hangouts and virtual game nights
✓ Content creators and streamers
✓ Fans of Choose Your Own Adventure and party games
✓ Anyone who loves great stories and friendly competition

Download now and discover why players are calling Story Clash "the most intense 15 minutes of my week."

Screenshots (10 total, designed for viral appeal):

1. Hero: Players in lobby with room code, "Gather Your Crew" tagline.
2. Minigame in action: dramatic moment, timer, intense visual.
3. Story scene: gripping text, choice buttons, countdown.
4. Turn indicator: "Your Turn" with player spotlight.
5. Genre reveal: dramatic card flip moment.
6. Ending screen: "You Survived" with player names.
7. Recap timeline: full story with choices highlighted.
8. Social proof: "Share Your Story" with mock tweet/post.
9. Genre variety: montage of 3 genre aesthetics.
10. "Play Again": lobby with returning players, emphasizing replayability.

App Preview Video (30 seconds):

1. (0-5s) Hook: "Your choices. Their fate. 15 minutes of chaos."
2. (5-10s) Lobby: friends joining, room code, excitement building.
3. (10-15s) Minigame: fast cuts, competitive action, leaderboard reveal.
4. (15-25s) Story phase: choice made, dramatic consequence, next player's turn, tension rising.
5. (25-30s) Ending: survival or doom, recap, "Play Again" CTA, app icon + download prompt.

Icon:

- Bold, recognizable at small sizes.
- Central image: stylized story book split open with glowing/glitched pages.
- Dark background, neon accent glow (cyan/red).
- No text in icon (clean, modern).

Category: Games > Word (primary), Games > Board (secondary).

Age Rating: 12+ (mild horror themes, no graphic violence).

---

8. Monetization & Growth Strategy

V1 (Launch):

- **Free:** 3 genres, unlimited play.
- **Premium Genre Packs:** $2.99 each (unlock 2 new genres per pack).
    - Pack 1: "Cosmic Horror" + "Cursed Island."
    - Pack 2: "Time Loop Mystery" + "Cyberpunk Heist."
- **No ads, no energy systems, no pay-to-win.**

V2 (3 months post-launch):

- **Story Vault Subscription:** $4.99/month.
    - Weekly exclusive story drops.
    - Early access to new genres.
    - Custom avatar cosmetics.
    - Priority matchmaking (future feature).
- **Cosmetic Packs:** $1.99 (avatar shapes, color themes).

Growth Tactics:

- **Viral Sharing:** Auto-generated story recaps optimized for Twitter/Instagram.
- **Streamer Kits:** Twitch/YouTube overlay graphics, OBS integration (future).
- **Referral Rewards:** Share room code → friend downloads → both unlock bonus cosmetic.
- **App Store featuring:** Target "Apps We Love," "New Games We Love," and holiday features.
- **Community:** Discord server, weekly story contests, player-submitted story ideas.

---

9. Success Metrics (V1 Launch Goals)

Week 1:

- 10,000 downloads.
- 3,000 active sessions.
- 4.5+ star rating (>100 reviews).
- <5% crash rate.

Month 1:

- 50,000 downloads.
- 20% DAU/MAU (strong retention).
- 15% conversion to premium genre pack.
- Featured in App Store "New Games We Love."

Month 3:

- 200,000 downloads.
- 5% subscription conversion.
- Influencer/streamer adoption (5+ creators with >10k followers).

---

10. Development Phases

Phase 1: Core Prototype (Week 1-2)

- Lobby system (create/join/sync).
- Basic minigame (Reflex Roulette).
- Story phase with 1 genre, 1 linear path, preset choices only.
- Minimal UI (functional, not polished).

Phase 2: V1 Feature Complete (Week 3-4)

- 3 full genres with branching paths.
- Free choice input with keyword mapping.
- Ending + recap screens.
- Full sound design + animations.
- UI polish to App Store standards.

Phase 3: Testing & Polish (Week 5)

- Internal playtesting (10+ sessions with real groups).
- Bug fixes, performance optimization.
- ASO assets (screenshots, video, copy).

Phase 4: Soft Launch (Week 6)

- TestFlight beta (50-100 testers).
- Collect feedback, iterate.

Phase 5: App Store Launch (Week 7)

- Submit to App Store.
- Press kit, social launch, influencer outreach.

Codex Prompts (High-Detail, Autonomous Execution)

Prompt 1: Project Scaffold

You are an expert full-stack developer building a multiplayer story game called "Story Clash" for the iOS App Store. The tech stack is: Next.js 14 (App Router), TypeScript, Tailwind CSS, Framer Motion, Socket.io (client + server), PostgreSQL via Supabase.

Create the complete project structure with:

1. Next.js app in `/app` directory with these routes:
   - / (home screen)
   - `/create` (create room)
   - `/join` (join room)
   - `/lobby/[code]` (lobby screen)
   - `/game/[code]` (story game session)
   - `/recap/[code]` (ending recap)
2. API routes in `/app/api`:
   - `/api/rooms/create` (POST: generates 4-char room code, returns room ID)
   - `/api/rooms/join` (POST: validates code, adds player, returns room data)
   - `/api/rooms/[code]` (GET: fetch room state)
3. Socket.io server setup:
   - Separate `/server` directory with Express + Socket.io.
   - Events: `join_room`, `leave_room`, `start_game`, `submit_choice`, `advance_turn`.
   - Room state management (players, turn order, current scene, story history).
4. Database schema (Supabase):
   - `rooms` table: id, code, created_at, expires_at, genre, active.
   - `players` table: id, room_id, name, avatar, score, turn_order.
   - `stories` table: id, genre, scenes (JSON).
   - `sessions` table: id, room_id, story_id, choices (JSON), ending.
5. Shared types in `/types`:
   - `Room`, `Player`, `Scene`, `Choice`, `StoryTree`, `SessionState`.
6. Utilities in `/lib`:
   - `generateRoomCode()` (4 random uppercase letters, no O/I).
   - `validateRoomCode(code: string)` (checks format + existence).
   - `mapFreeChoiceToPath(input: string, keywords: object)` (keyword matching).
7. Initial styling:
   - Tailwind config with custom colors from design system (charcoal, cyan, red, green).
   - Global CSS with font imports (Poppins Bold, Inter).
8. Package.json with scripts:
   - `dev` (runs Next.js + Socket.io server concurrently).
   - `build`, `start`, `lint`.

Generate all files with complete boilerplate code, TypeScript types, and inline comments explaining each module's purpose. Use modern React patterns (server components where appropriate, client components with 'use client' for interactivity).

Prompt 2: Lobby System

Implement the complete lobby system for Story Clash.

Requirements:

1. **Home Screen (/app/page.tsx):**
   - Dark background (#1a1a1d).
   - Centered hero text: "Story Clash" with animated glitch effect using Framer Motion (subtle x/y jitter, 2s loop).
   - Two large buttons:
     - "Create Room" (cyan glow border, navigates to `/create`).
     - "Join Room" (subtle pulse animation, navigates to `/join`).
   - "How to Play" link at bottom (opens modal with placeholder text for now).
   - Animated particle background (use CSS or Framer Motion for drifting abstract shapes).
2. **Create Room Flow (/app/create/page.tsx):**
   - Input: player display name (max 12 chars, trim whitespace, basic profanity filter array check).
   - On submit:
     - Call `/api/rooms/create` (POST with player name).
     - Receive room code + room ID.
     - Navigate to `/lobby/[code]`.
   - Loading state during API call (spinner + "Creating room...").
3. **Join Room Flow (/app/join/page.tsx):**
   - Two inputs: room code (4 chars, auto-uppercase as user types) + player name.
   - Real-time validation: as code is typed, show green checkmark if valid format.
   - On submit:
     - Call `/api/rooms/join` (POST with code + name).
     - On success: navigate to `/lobby/[code]`.
     - On error: show inline error message ("Room not found" or "Room is full").
4. **Lobby Screen (/app/lobby/[code]/page.tsx):**
   - Header: display room code large and bold, with "Copy Code" button (copies to clipboard, shows "Copied!" toast for 2s).
   - Player list:
     - Fetch room state on mount via `/api/rooms/[code]`.
     - Real-time updates via Socket.io (`player_joined`, `player_left` events).
     - Display each player: avatar (colored circle with first initial), name, host crown icon for first player.
     - Empty slots show "Waiting for players..." (gray placeholder).
   - "Share Room" button: generates shareable text ("Join my Story Clash game! Code: [CODE]") and copies to clipboard.
   - "Start Game" button (only visible to host):
     - Disabled if <3 players (gray, with tooltip "Need 3+ players").
     - Enabled if 3-6 players (cyan glow).
     - On click: emit `start_game` socket event, navigate all players to `/game/[code]`.
   - Ambient sound: use `<audio>` tag with lobby ambient drone (placeholder: any CC0 drone sound, loop, low volume).
   - Disconnect handling: on unmount, emit `leave_room` event.

5. **API Routes:**
    - Implement /api/rooms/create: generate code, insert into Supabase rooms + players tables, return { code, roomId }.
    - Implement /api/rooms/join: validate code, check room capacity (max 6), insert player, return room data.
    - Implement /api/rooms/[code]: fetch room + players, return as JSON.
6. **Socket.io Events (server-side in /server):**
    - join_room: add player to room state, broadcast player_joined to room.
    - leave_room: remove player, broadcast player_left.
    - start_game: validate host, broadcast game_started with initial game state.

Styling:

- Use Tailwind for all layout and colors (follow design system from spec).
- Framer Motion for animations (button hovers, page transitions, glitch effect).
- Mobile-responsive (single-column layout, large touch targets).

Testing:

- Add console logs for all socket events.
- Handle edge cases: duplicate names (append number), room expiration (return error if expired).

Generate all code with complete implementations, no TODOs. Include error handling and loading states.

Prompt 3: Minigame (Reflex Roulette)

Implement the "Reflex Roulette" minigame for Story Clash. This runs after "Start Game" is clicked in the lobby.

Requirements:

1. **New Route: /app/minigame/[code]/page.tsx**
    - Redirected here when game starts.
    - All players see the same synchronized minigame.
2. **Countdown Phase (3 seconds):**
    - Full-screen countdown: "3", "2", "1", "GO!" (each 1 second).
    - Large bold text (80px), centered, with scale-in animation (Framer Motion).
    - Sound effect on each count (use HTML5 Audio, placeholder beep sounds).
3. **Minigame Phase (3 rounds, 5 seconds each):**
    - Display circular target (300px diameter, centered):
        - Outer circle: dark gray border.
        - "Hit zone": 60-degree colored arc (cyan) rotating clockwise (2s per rotation).
        - Needle: thin line from center, rotating counter-clockwise (1.5s per rotation).
    - Player taps/clicks screen when needle is inside hit zone.
    - On tap:
        - Calculate accuracy: distance of needle from center of hit zone (0-100 scale).

        - Calculate speed bonus: earlier taps in round = higher multiplier (1.0x-1.5x).
        - Display score for that round (fade in "+85 pts" above target, then fade out).
        - Emit `minigame_score` socket event with round score.
    - After 3 rounds: wait for all players to finish (show "Waiting for others..." for fast players).
4. **Results Phase:**
    - Fetch all player scores from server.
    - Display leaderboard:
        - Animated slide-in, sorted by total score (highest first).
        - Each row: player name, avatar, total score.
        - Top player highlighted with "Story Master" badge (gold crown icon).
    - Assign turn order = leaderboard rank.
    - Genre selection (only for top player):
        - Show 3 face-down genre cards (mystery backs).
        - Top player taps one card.
        - Card flips with 3D rotation animation (Framer Motion).
        - Reveal genre name + thematic icon (zombie, alien, haunted house).
        - Full-screen genre announcement ("Zombie Outbreak!", dramatic sound effect, 3s display).
        - Emit `genre_selected` event with chosen genre.
    - All other players see "Story Master is choosing genre..." then see the reveal simultaneously.
5. **Transition to Story:**
    - After genre reveal, auto-navigate all players to `/game/[code]` after 2s delay.
6. **Socket Events:**
    - `minigame_start`: server broadcasts to sync countdown start time.
    - `minigame_score`: client sends { playerId, round, score, accuracy }.
    - `minigame_complete`: server calculates leaderboard, broadcasts { players, turnOrder }.
    - `genre_selected`: server broadcasts { genre }.
7. **Visuals:**
    - Use HTML5 Canvas or SVG for target/needle (60fps animation).
    - Rotating elements use `requestAnimationFrame` for smooth motion.
    - Mobile-optimized: full-screen tap area (entire screen is tappable, not just target).
8. **Accessibility:**
    - Visual cue when needle enters hit zone (hit zone pulses briefly).
    - Haptic feedback on tap (iOS vibration API).

Genres (hardcoded for V1):
[
{ "id": "zombie", "name": "Zombie Outbreak", "icon": "□" },
{ "id": "alien", "name": "Alien Invasion", "icon": "□" },
{ "id": "haunted", "name": "Haunted Manor", "icon": "□" }
]

Generate complete implementation with animations, socket sync, and error handling. No TODOs. Optimize for mobile touch responsiveness.

Prompt 4: Story Phase (Turn-Based Gameplay)

Implement the core story phase gameplay for Story Clash at `/app/game/[code]/page.tsx`.

Requirements:

1. **Game State Management:**
    - Fetch initial game state on mount: { genre, currentScene, turnOrder, activePlayerIndex, history }.
    - Subscribe to socket events: `scene_update`, `turn_advance`, `game_end`.
    - Local state: currentScene (text + choices), activePlayer, timer, historyLog.
2. **Scene Display (All Players):**
    - Full-screen dark background with genre-specific subtle animated overlay:
        - Zombie: flickering shadow vignette.
        - Alien: scanline glitch effect.
        - Haunted: slow fog drift.
    - Story text area (center, 60% width):
        - Display `currentScene.text` with typewriter animation (30 chars/sec, Framer Motion or custom JS).
        - Text: white, 18px, line-height 1.6, readable serif font (e.g., Merriweather or similar).
    - Turn indicator banner (top): "Your Turn, [Name]" if active player, or "Waiting for [Name]..." if spectating.
    - Player order sidebar (right, 20% width):
        - List all players in turn order.
        - Highlight active player (cyan glow border).
        - Show small avatar + name for each.
3. **Active Player UI:**
    - Countdown timer (30 seconds):
        - Circular progress bar shrinking clockwise (green → yellow at 15s → red at 10s).
        - Centered above choices, with numeric countdown inside (e.g., "18").
        - On timeout: auto-select random preset choice, emit `choice_timeout` event.
    - Choice buttons (two preset choices):
        - Large buttons (full width of text area, stacked vertically, 60px height).
        - Text: choice from `currentScene.choices[].text`.
        - Hover: subtle scale + glow effect.
        - On click: emit `submit_choice` event with { playerId, choiceId }, disable buttons, show "Submitting..." state.
    - Free choice option:
        - Below preset buttons: "Or describe your own action..."
        - Text input (max 60 chars, real-time char counter).
        - Submit button (only enabled when text length > 5 chars).
        - On submit: emit `submit_choice` with { playerId, freeText }.
4. **Spectating Players UI:**
    - See same scene text and turn indicator.
    - Choices area shows: "Waiting for [ActivePlayer] to decide..." with animated ellipsis.
    - No interactive elements.
5. **Choice Resolution:**

   - Server receives choice, determines next scene using story tree logic (see below).
   - Server emits `scene_update` event with { nextScene, choice made, activePlayer }.
   - Client: brief suspense beat (1s screen darken + sound cue), then display new scene with typewriter effect.
   - Advance turn to next player in order.
6. **Story Tree Logic (Server-Side):**
   - Load story data from database or JSON file based on genre.
   - Each scene has:
     ```
     {
     "id": "scene_01",
     "text": "You wake in a dark room. Glass shatters upstairs.",
     "choices": [
     { "id": "choice_a", "text": "Search for a weapon", "next": "scene_02a" },
     { "id": "choice_b", "text": "Find an exit", "next": "scene_02b" }
     ],
     "freeChoiceKeywords": {
     "weapon|arm|fight": "scene_02a",
     "exit|leave|run": "scene_02b",
     "default": "scene_02b"
     }
     }
     ```
   - For free choice: use `mapFreeChoiceToPath()` utility (keyword matching, case-insensitive, split input into words, check against keyword patterns).
   - Track choice history in session state (for recap).
7. **Game End Condition:**
   - After 8-12 scenes (determined by story tree), reach an ending node.
   - Ending node has `"ending": true` flag and `"endingType": "triumph" | "survival" | "doom"`.
   - Server emits `game_end` event with { endingScene, endingType, history }.
   - Auto-navigate all players to `/recap/[code]`.
8. **Sound Design:**
   - Ambient genre soundscape (loop, low volume):
     - Zombie: distant groans + wind.
     - Alien: static + mechanical hum.
     - Haunted: whispers + creaking.
   - UI sounds:
     - Button click: soft tap.
     - Choice submit: swoosh.
     - Timer warning (last 10s): heartbeat rhythm.
     - Scene transition: suspense sting.
   - Use HTML5 Audio, preload all sounds, provide mute toggle (settings icon, top-right).
9. **Edge Cases:**
   - Player disconnect: skip their turn after 10s, continue game.
   - Reconnect: rejoin at current scene, catch up on history (show "Reconnected" toast).

Sample Story Data (Zombie Genre, 3 scenes for testing):

```
{
```

"genre": "zombie",
"scenes": [
{
"id": "start",
"text": "You wake in a dark room. The smell of rot fills the air. Glass shatters upstairs.",
"choices": [
{ "id": "a", "text": "Search for a weapon", "next": "armed" },
{ "id": "b", "text": "Try to find an exit", "next": "exit_attempt" }
],
"freeChoiceKeywords": {
"weapon|arm|knife|gun": "armed",
"exit|door|leave|run": "exit_attempt",
"default": "exit_attempt"
}
},
{
"id": "armed",
"text": "You find a crowbar under debris. Groans echo from the hallway. You're not alone.",
"choices": [
{ "id": "a", "text": "Confront the threat", "next": "ending_survival" },
{ "id": "b", "text": "Sneak past quietly", "next": "ending_triumph" }
],
"freeChoiceKeywords": {
"fight|attack|confront": "ending_survival",
"sneak|hide|quiet": "ending_triumph",
"default": "ending_survival"
}
},
{
"id": "exit_attempt",
"text": "The door is barricaded. Something pounds against it from the other side. You hear screams outside.",
"choices": [
{ "id": "a", "text": "Break through", "next": "ending_doom" },
{ "id": "b", "text": "Find another way", "next": "armed" }
],
"freeChoiceKeywords": {
"break|force|open": "ending_doom",
"find|search|other": "armed",
"default": "ending_doom"
}
},
{
"id": "ending_triumph",
"text": "You slip past the undead and escape into the dawn. You survived. For now.",
"ending": true,
"endingType": "triumph"
},
{
"id": "ending_survival",
"text": "You fight through the horde. Exhausted and wounded, but alive.",

"ending": true,
"endingType": "survival"
},
{
"id": "ending_doom",
"text": "The door gives way. The dead pour in. Darkness takes you.",
"ending": true,
"endingType": "doom"
}
]
}

Implement complete story phase with all UI, animations, socket sync, and story logic. No placeholders. Handle all edge cases. Optimize for 60fps on mobile.

Prompt 5: Ending & Recap Screen

Implement the ending and recap screen for Story Clash at `/app/recap/[code]/page.tsx`.

Requirements:

1. **Ending Reveal:**
   - On mount, fetch game session data: { endingScene, endingType, history }.
   - Full-screen ending scene display:
     - Background: theme-specific visual (triumph = bright dawn, survival = flickering light, doom = fade to black).
     - Display `endingScene.text` with slow typewriter effect (20 chars/sec).
     - Ending type badge (large, centered below text):
       - Triumph: "Victory" (gold, glowing).
       - Survival: "Narrowly Escaped" (yellow, pulsing).
       - Doom: "Game Over" (red, flickering).
     - Sound: ending-specific music (triumph = uplifting, doom = ominous drone).
     - Display for 5 seconds, then auto-scroll to recap.
2. **Recap Timeline:**
   - Scrollable vertical timeline showing all scenes + choices.
   - Each timeline node:
     - Scene number + brief text snippet (first 50 chars + "...").
     - Player who chose + their avatar.
     - Choice made (highlight in cyan if preset, green if free choice).
   - Key turning points highlighted with icon (e.g., "Critical moment!" badge on choices that led to ending).
   - MVP (Most Valuable Player) callout:
     - Determine MVP based on predefined logic (e.g., player who made choice leading to best ending, or most creative free choices).
     - Display at bottom of timeline: "[Player] was MVP!" with crown icon and brief reason ("Made the save" or "Creative thinker").
3. **Action Buttons:**
   - "Play Again" (primary CTA, cyan, large):
     - On click: emit `restart_session` socket event.
     - Navigate all players back to `/lobby/[code]` with same players.

    - "Share Story" (secondary, white outline):
        - Generate shareable text:
          We just played Story Clash!
          Story: [Genre Name]
          Ending: [Ending Type]
          MVP: [Player Name]
          Join us: [App Store Link]
        - Copy to clipboard, show "Copied! Share with friends" toast.
        - Future: generate image with Open Graph meta tags for social sharing.
    - "Exit" (small text link, bottom):
        - Navigate to home page (/).
4. **Visual Design:**
    - Dark background with subtle gradient (top = genre color accent, bottom = charcoal).
    - Timeline: vertical line (2px cyan) connecting nodes.
    - Nodes: cards with soft shadow, white text, player avatar inset.
    - Smooth scroll with snap-to-node behavior (CSS scroll-snap).
    - Animated entry: timeline nodes fade in sequentially (50ms stagger, Framer Motion).
5. **Data Structure (Session History):**
   {
   "endingScene": { "text": "...", "endingType": "triumph" },
   "history": [
   { "sceneId": "start", "sceneText": "You wake in...", "player": "Alice", "choice": "Search for weapon", "isFreeChoice": false },
   { "sceneId": "armed", "sceneText": "You find a crowbar...", "player": "Bob", "choice": "sneak quietly", "isFreeChoice": true }
   ],
   "mvp": { "player": "Alice", "reason": "Made the save" }
   }
6. **Accessibility:**
    - Keyboard navigation for buttons.
    - Screen reader support for timeline ("Scene 1, Alice chose: Search for weapon").

Generate complete recap screen with animations, data handling, and social sharing. No TODOs. Ensure mobile responsiveness and smooth scrolling.

Prompt 6: Polish, Sound Design, and Final Testing

Add final polish and production-readiness to Story Clash.

Tasks:

1. **Sound System:**
    - Create a centralized sound manager (/lib/soundManager.ts) with:
        - preloadSounds(soundUrls[]): load all audio files on app init.
        - play(soundId, options): play sound with volume control, loop, fade-in.
        - stop(soundId): stop and reset sound.
        - setMasterVolume(volume): global volume control.
        - mute() / unmute(): toggle all sounds.
    - Integrate into all pages:

        - Lobby: ambient drone (loop).
        - Minigame: countdown beeps, tap sound, results sting.
        - Story: genre soundscapes (loop), UI clicks, timer warning, suspense sting.
        - Recap: ending music.
    - Add persistent mute toggle (settings icon in top-right corner, all pages, state saved in localStorage).
2. **Animations & Transitions:**
    - Page transitions: fade + slide (200ms, Framer Motion `<AnimatePresence>`).
    - Button interactions: scale(1.05) on hover, scale(0.95) on click.
    - Story text: smooth typewriter effect with cursor blink.
    - Minigame target: 60fps rotation (use `transform: rotate()` with CSS animations or `requestAnimationFrame`).
    - Recap timeline: staggered fade-in (Framer Motion `variants`).
3. **Accessibility Enhancements:**
    - Add ARIA labels to all interactive elements.
    - Ensure all text has minimum 4.5:1 contrast ratio (test with WebAIM tool).
    - Keyboard navigation: tab order logical, focus indicators visible (cyan outline).
    - Haptic feedback on iOS: use `navigator.vibrate()` for key actions (choice submit, minigame tap).
    - Screen reader testing: ensure story text and choices are announced correctly.
4. **Performance Optimization:**
    - Lazy load routes (Next.js dynamic imports).
    - Optimize images: use WebP format, lazy loading with `<Image>` component.
    - Minify audio files (use Audacity or online tools to compress to ~128kbps).
    - Socket.io: batch updates (emit at most once per 100ms for real-time sync).
    - Lighthouse audit: target 90+ performance score on mobile.
5. **Error Handling & Edge Cases:**
    - Network errors: show toast with "Connection lost. Reconnecting..." (retry logic with exponential backoff).
    - Room expiration: if player joins expired room, redirect to home with error message.
    - Player disconnect during game: skip turn after 10s timeout, show "[Player] disconnected" message.
    - Graceful degradation: if WebSockets fail, fall back to HTTP polling (show "Limited connectivity" warning).
6. **Testing Checklist:**
    - Test full game flow with 3, 4, 5, and 6 players.
    - Test all 3 genres, all ending types.
    - Test free choice keyword matching with edge cases (typos, unexpected words).
    - Test on iOS Safari, Chrome mobile, and desktop Chrome.
    - Test with poor network (throttle to 3G in DevTools).
    - Test accessibility: keyboard-only navigation, screen reader (VoiceOver on iOS).
7. **App Store Preparation:**
    - Generate app icon (1024×1024, PNG, per design spec).
    - Create 10 screenshots (iPhone 15 Pro Max resolution: 1290×2796).
    - Record 30s app preview video (use screen recording + editing tool).
    - Write App Store description, keywords, subtitle (per ASO spec above).
    - Set up TestFlight for beta testing (upload build, invite 50 testers).
8. **Final Code Review:**
    - Remove all console.logs (except critical errors).

    - Add inline comments for complex logic (story tree mapping, socket event handling).
    - Ensure TypeScript strict mode enabled, no `any` types.
    - Run `npm run lint`, fix all warnings.
    - Run Prettier for consistent formatting.

Generate all polish updates, sound integration, and testing utilities. Provide a final checklist for production deployment.

Summary for Codex Execution

This specification is designed for **autonomous Codex execution** with minimal human intervention. Each prompt is:

- **Self-contained:** All context, requirements, and examples included.
- **Explicit:** No ambiguity in features, UI, or technical implementation.
- **Production-ready:** Targets App Store #1 quality with polished UX, accessibility, and performance.

Execution Strategy:

1. Run prompts sequentially (1 → 6).
2. After each prompt, review generated code for completeness (should have zero TODOs).
3. Test each phase before proceeding (scaffold → lobby → minigame → story → recap → polish).
4. Iterate only on bugs or edge cases discovered during testing.

Expected Timeline:

- Prompts 1-2: Week 1 (scaffold + lobby).
- Prompts 3-4: Week 2-3 (minigame + story phase).
- Prompt 5: Week 4 (recap + ending).
- Prompt 6: Week 5 (polish + testing).

Result: A complete, App Store-ready multiplayer narrative game built largely by Codex, with human oversight on design decisions, testing, and final polish.