.

Here's a \*\*complete, ready-to-publish Markdown specification\*\* that you can use on GitHub or in documentation for your frontend development team.

---

# 🎹 Scaler3 AI Chord Generator – Frontend Specification (Sprint 1)

This document outlines the \*\*requirements and implementation plan\*\* for building a \*\*fully working, testable frontend interface\*\* for a web-based clone of Scaler 3's AI Chord Generator. This is part of \*\*Sprint 1\*\*, focusing on UI development, mock data generation, and core playback functionality.

---

## 🎯 Goal

As a web developer, your task is to build the \*\*frontend interface\*\* that allows users to:

- Select musical parameters (key, scale, chord type)

- Set progression length (number of bars)

- Generate chords using mock functionality

- View results in a piano roll canvas

- Play, pause, and stop playback using Web Audio API

---

## 📋 Requirements

### \*\*User Interface\*\*

Create a responsive layout with two main sections:

#### Sidebar

- Dropdowns for:

- Key (C, C#, D, ..., B)

- Scale (major, minor, pentatonic, etc.)

- Chord Type (triad, seventh, augmented, etc.)

- Number input for progression length (1–16 bars)

- BPM input (default: 85 BPM)

- Time signature selection (e.g., 3/4, 4/4)

#### Main Area

- Canvas element for piano roll visualization

- Playback controls (`Play`, `Pause`, `Stop`)

- Virtual piano aligned vertically with MIDI notes on the left side of the piano roll view

- Buttons for:

- Record MIDI Notes (from external devices like BOSS GM-800)

- Play MIDI Notes (to external devices like Arturia Minifreak)

- Save MIDI Notes or settings to JSON file

- Load MIDI Notes or settings from JSON file

---

### \*\*JavaScript Logic\*\*

- Attach event listeners to all inputs and buttons

- Implement a mock function that generates dummy MIDI notes as if from an AI model

- Render generated chords on the piano roll canvas

- Use Web Audio API to play simple sine-wave tones when “Play” is clicked

- Start a metronome ticking at the set BPM when playback starts

---

### \*\*Responsive Design\*\*

- Ensure the layout works well on desktop and mobile devices

- Use CSS Flexbox or Grid for layout structure

---

### \*\*📁 File Structure\*\*

```

frontend/

│

├── index.html # Main HTML page

├── styles.css # Styling for the app

└── app.js # JavaScript logic

```

---

## ✅ Acceptance Criteria

- [x] User can select key (C, C#, D, ..., B)

- [x] User can select scale (major, minor, pentatonic, etc.)

- [x] User can select chord type (triad, seventh, augmented, etc.)

- [x] User can set progression length (1–16 bars)

- [x] Clicking "Generate Chords" triggers a function and displays output

- [x] Piano roll draws rectangles representing notes over time

- [x] Play/Pause/Stop buttons control audio playback of generated chords

- [x] No backend calls are made yet; all data is mocked

- [x] All code must run locally without external dependencies

---

## 🧪 Testing Plan

- Manual testing by QA or developer to ensure UI responsiveness and interaction

- Test edge cases like invalid input (ensure graceful handling)

- Check cross-browser compatibility (Chrome, Firefox, Safari)

- Ensure mobile view works correctly

---

## 🔧 Tools & Technologies

- \*\*VS Code\*\* – IDE for development

- \*\*Live Server Extension\*\* – For local testing

- \*\*Git\*\* – Version control (commit early and often)

- \*\*Browser DevTools\*\* – Debugging JavaScript and layout issues

---

## 📦 Deliverables

- Fully functional `index.html`, `styles.css`, and `app.js` files

- Demo-ready build that runs locally

- `README.md` explaining how to run and test the frontend

- Optional: Unit tests for core functions (e.g., chord drawing logic)

---

## 🧾 UUID

`UUID: 1f5d7c4e-bb8c-4e4d-a1d1-dba8989323f9`

---

Would you like me to generate the actual `index.html`, `styles.css`, and `app.js` files next? Or package them into a downloadable ZIP?.