Web Application Requirements

Overview

Develop a single-page web application that simulates a music store showcase by generating fake song information.

Functional Requirements

LANGUAGE SELECTION

- Must include English (USA).
- Include at least one additional language/region (e.g., German (Germany) or Ukrainian (Ukraine)).

SEED CONFIGURATION

- Allow the user to set a custom 64-bit seed value.
- Provide an option to generate a random seed.

LIKES PER SONG

- The user specifies the average number of likes per song.
- Range: 0–10, with fractional values allowed (e.g., 3.7).
- Examples:
  - 0 — no likes for any song.
  - 0.5 — each song has either 1 like or 0 likes with 1:1 chances (on average, 1 like for every 2 songs).
  - 10 — each song has exactly 10 likes.
- Fractional values must be implemented probabilistically to achieve the desired average.

UI / UX

- The controls for language selection, seed configuration, and likes-per-song should be placed horizontally in a single row, functioning as a toolbar.
- Data must update dynamically whenever parameters change, without requiring the user to press Enter or click separate buttons.
- Support two display modes:
  - **Table View** with pagination (no infinite scrolling).
  - Gallery View with infinite scrolling (no pagination; in the further text, a "batch" of data in Gallery View is also called a page).
- Any change to generation parameters must:
  - Reset the Table View to the first page.
  - Reset the Gallery View to the initial scroll position.
- Data should be provided to the user immediately upon page load, without requiring any input or button presses.

GENERATED DATA

Each table row and each gallery card must contain the following data:

- Sequence index — a sequential number starting from 1 (e.g., 1, 2, 3, …).
- Song title — randomly generated.
- Artist — randomly generated, with a mix of band names and personal names.
- Album title — randomly generated, or the literal string "Single".
- Genre — randomly generated.

AUTHENTICATON

The application must function without requiring user registration or authentication.

LANGUAGE INDEPENDENCE

- You are not required to translate the same song between languages.
- Songs for different languages must be generated independently.

LOCALIZATION RULES

- The language of the song title, artist name, album title, and genre must match the selected language/region.
- Generated data should look realistic for the chosen language/regionl (although can be nonsensical, e.g., "Fast Carrot Plus" is OK).
- Placeholder text such as lorem ipsum is not acceptable.

PARAMETER INDEPENDENCE

- All parameters — region, seed, and likes — are independent.
- Changing one parameter does not reset the others, but:
  - Changing likes only updates the like counts, titles, artists, etc. stay the same.
  - Changing seed or region changes titles, artists, albums, genres, and covers.
- Likes, titles, artists, and albums must update dynamically in response to parameter changes.

TABLE VIEW

Each record in the Table View must be expandable. The expanded view should be collapsible so the table returns to its original state without a full page reload.
When the user clicks on a record, detailed information is displayed, including:

- Album or single cover image — randomly generated, with:
  - The correct title rendered on the cover.
  - The correct artist name rendered on the cover.
  - A randomized background image or pattern.
- Play button — allows playback of a short preview clip
- Some randomly generated review text.

SONG GENERATION

- You must generate actual random music that can be played directly in the browser.
- Songs must be reproducible: the same seed values must always generate the same audio output.

OPTIONAL REQUIREMENTS

- Add an Export button that generates a ZIP archive containing MP3 files. Each file should be named using the corresponding song title, album title, and artist name.
- Implement lyrics display with live scrolling synchronized to the playback of the "song".

Implementation Notes

REGION-SPECIFIC DATA

Your code must not contain any hardcoded region-specific values, words, or other data. All locale-specific content must come from external resources (e.g., lookup tables, configuration files, or localization datasets) so that languages and regions can be added, removed, or updated without modifying the source code.

ARCHITECTURE

You do not store random data on the server, but all data must be generated on the server side, not in the browser. The browser connects to a single server that provides a single page (a batch) of data generated in memory. No database is required for storing random data (though you may use a database for lookup tables, however it is generally not a best approach in this case).

SEEDS

When the user changes the seed, the generated data must update accordingly. The seed provided to the RNG algorithm should be a combination of the user-specified seed and the displayed page number. The exact method of combining the seed and page number is not critical; a simple MAD operation is sufficient. **Important**: the same seed, when entered at any time, must produce identical data, ensuring reproducibility across devices and dates.

Changing the seed must change the generated data, but the same seed must always produce the same results. Adjusting the number of likes must not alter generated titles, artists, or other core content—these should depend solely on the seed value and the record index.

CODE REUSE

You are required to use third-party libraries for random data generation (e.g., some port of Faker) and **may need to extend these libraries** in ways supported by their APIs. It is expected that you will also use appropriate libraries for tasks such as music generation, implementing infinite scrolling, and other required functionality.

Additional notes

- Data should look as realistic as possible, though it does not need to be semantically meaningful.
- Note that higher grades will be given to titles and covers that appear visually appealing and realistic, as well as songs that sound like coherent melodies and rhythms rather than random sequences of pitches.
- You are encouraged to use libraries that incorporate music theory, such as randomly selecting chord progressions for verses and choruses, splitting notes into separate tracks with different instruments, adjusting tempo, rendering MIDI files, and applying audio effects like reverberation and echo.
- AI services may also be used to enhance the generation if desired. If you enjoy working with AI agents, feel free to leverage them!
- Keep in mind this is only a small part of the task, so manage your expectations and the time spent accordingly. At minimum, a simple implementation generating random sequences of notes is acceptable.