**Overview**

The **ReviewPage** component renders a two-step review interface:

* **For non-kanji cards (vocab/grammar):**  
  A single-step flow is used where the user types the meaning and then immediately submits.
* **For kanji cards:**  
  A two-step flow is used. First, the user enters the meaning, and after submission, if the card is kanji the interface advances to a reading step. Only after both steps are completed is the final correctness computed.

For guest users, the page updates progress in the browser’s sessionStorage using the same SRS (spaced repetition system) logic as the backend.

**Dependencies**

* **React Hooks:**  
  useState, useEffect, useRef, and useCallback are used for state management, side effects, and performance optimization.
* **Next.js Router:**  
  useRouter is used for navigation and reading query parameters (e.g., to set the review mode).
* **Custom Components:**
  + Header – renders the app header.
  + Flashcard – displays an individual flashcard.
* **Third-party libraries:**
  + wanakana – used to convert user input (romaji to hiragana).
  + damerau-levenshtein – used to calculate the edit distance between the user’s answer and the accepted answers.
  + luxon (via the backend srs.js file) – used in the SRS logic to compute next review times.
* **SRS Logic:**  
  The function getNextReviewDateAndLevel is imported from the backend utilities (backend/utils/srs) and is used in the guest SRS update logic.

**Helper Functions**

**getTypeStyles(type)**

Returns a style object with a background and text color based on the flashcard type.

* **Parameters:**
  + type (string): The type of flashcard (e.g., "vocab", "grammar", "kanji").
* **Returns:**
  + An object containing bg and text classes.

**removeBrackets(str)**

Removes any text within parentheses from a string.

* **Parameters:**
  + str (string): The string to process.
* **Returns:**
  + A new string with bracketed text removed.

**containsKanji(str)**

Checks whether a string contains Kanji characters using a regular expression.

* **Parameters:**
  + str (string): The input string.
* **Returns:**
  + true if Kanji is found; otherwise, false.

**localGuestSRSUpdate(flashcardId, bothCorrect)**

Handles the guest user’s SRS (spaced repetition system) update locally using sessionStorage. This function:

1. Reads the previous progress (level, correct count, incorrect count, next review, reached\_level\_3) for the given flashcard from sessionStorage.
2. Uses getNextReviewDateAndLevel (imported from the backend utilities) to compute the new level and next review timestamp based on the current level and whether the answer was completely correct (bothCorrect).
3. Increments the correct\_count if bothCorrect is true; otherwise, increments the incorrect\_count.
4. Updates the entry in sessionStorage.

* **Parameters:**
  + flashcardId (string | number): The ID of the flashcard.
  + bothCorrect (boolean): Indicates whether the user answered both steps correctly (or, for single-step cards, that the answer was correct).

**The ReviewPage Component**

**State Variables**

* **Flashcard Data & Navigation:**
  + flashcards: List of flashcards fetched from the backend.
  + currentIndex: The index of the current flashcard.
  + step: For Kanji cards, holds the current review step ("meaning" or "reading").
* **User Input:**
  + userMeaning: The text the user types for the meaning.
  + userReading: The text the user types for the reading.
* **Correctness States:**
  + meaningIsCorrect: Boolean indicating if the meaning answer is correct.
  + readingIsCorrect: Boolean indicating if the reading answer is correct (for Kanji).
* **Submission & Loading:**
  + showResult: Whether to show the result (correct/wrong).
  + submittable: To prevent multiple submissions.
  + loading and error: For handling API loading and errors.
* **User Identification:**
  + userId: Pulled from localStorage; if null, the user is treated as a guest.
* **UI Toggles:**
  + showFurigana: Toggles whether furigana is displayed.
  + showFullFlashcard: Toggles a detailed view of the flashcard.
* **References:**
  + meaningInputRef and readingInputRef: For auto-focusing the correct input field.

**Component Lifecycle**

1. **On Mount:**
   * Reads user\_id from localStorage.
2. **Fetching Flashcards:**
   * Once the router is ready (and userId is known), the component fetches review flashcards from the backend.
   * For guest users, it filters out any cards already at or above level 3 from sessionStorage.
3. **Handling User Input & Submission:**
   * The component provides input handlers for meaning and reading.
   * handleSubmit:
     + For non-kanji cards, it computes correctness and, if the answer is correct, calls the backend (or local guest update) with bothCorrect set to isCorrect.
     + For Kanji cards, the reading step is handled separately after the meaning step.
   * handleNext:
     + For Kanji cards:
       - On the “meaning” step, it moves to the “reading” step without resetting meaningIsCorrect so that the final evaluation (both steps) can be computed.
       - On the “reading” step, it computes bothCorrect as meaningIsCorrect && readingIsCorrect.
       - For single-step cards (vocab/grammar), it re-queues the card if the answer was not correct.
4. **Global Keydown & Auto-focus:**
   * The component listens for the Enter key to trigger submission or moving to the next card.
   * It automatically focuses the correct input field depending on the current step.

**Sub-Components**

**MeaningStep**

Renders the input field for the meaning answer.

* **Props:**
  + inputRef: A React ref for the input element.
  + userMeaning: The current user input.
  + onMeaningChange: Callback when the input changes.
  + showResult: Whether to display the result.
  + correctMeaning: The correct meaning (for display if the answer is wrong).
  + meaningIsCorrect: Boolean indicating if the answer is correct.

**ReadingStep**

Renders the input field for the reading answer (used only for Kanji cards).

* **Props:**
  + inputRef: A React ref for the input element.
  + userReading: The current user input.
  + onReadingChange: Callback when the input changes.
  + showResult: Whether to display the result.
  + correctReading: The correct reading (for display if the answer is wrong).
  + readingIsCorrect: Boolean indicating if the answer is correct.

**ResultDisplay**

Renders a result message for the current step based on whether the answer was correct or wrong.

* **Props:**
  + step: Indicates whether it’s the “meaning” or “reading” step.
  + meaningIsCorrect: Boolean for the meaning result.
  + readingIsCorrect: Boolean for the reading result.

**Debugging and Logging**

Throughout the code, several [DEBUG] console.log statements have been added to track:

* When the user ID is read from localStorage.
* The payload used for fetching flashcards.
* The raw data returned from the review endpoint.
* Filtering of flashcards for guest users (showing each card’s ID and its progress entry).
* In handleSubmit for both meaning and reading steps, logging the final computed correctness.
* In localGuestSRSUpdate, logging the old level, new level, next review date, and which count (correct or incorrect) is being incremented.
* In handleNext, logging the transition between steps and whether the card is being re-appended.

Review these logs in your browser’s console (using DevTools) to determine if the values you compute match your expectations.

**Conclusion**

This documentation covers the structure, functions, and debugging strategy for your review.js file. Use this document as a reference when modifying or debugging the immediate review flow, especially when working with guest user progress and ensuring that the SRS logic is correctly applied.

Feel free to adjust or extend this documentation to suit your project's needs.