

## **Specification: Letter-Sequence Reading Trainer**

### **1. Purpose**

A browser-based tool that helps children learn to read by constructing words letter-by-letter from a word list. The interface progressively filters available letters based on what can follow the current sequence. The app also incorporates phonetic letter sounds, speech recognition for reading aloud, and a log of completed words.

---

### **2. User Workflow**

#### **1. Initial Screen**

- Large empty sequence display.
- Full alphabet grid.

#### **2. Selecting a Letter**

- Letter appears at the top.
- The phonetic sound of the letter plays immediately.
- The grid updates to show only valid next letters.

#### **3. Building a Word**

- User continues selecting letters.
- If the current sequence matches any complete word in the list:
  - Display highlights the word as complete.
  - A button appears: “Read this word”.

#### **4. Reading the Word (Speech Recognition)**

- When the user presses “Read this word”:
  - App activates speech recognition.
  - User says the word aloud.
  - If the spoken result matches the target word:
    - Confetti animation plays.
    - The word is added to a “Completed Words Log”.
    - A prompt appears: “Start a new word?”
  - If incorrect:
    - A gentle audio cue indicates mismatch.

#### **5. Backtracking**

- “Back” removes the last letter.
- Grid updates accordingly.

#### **6. Resetting**

- “Clear” clears the entire sequence and restores the full alphabet.

## 7. Settings View

- Text area for editing the word list.
  - Text area showing completed words.
  - Option to clear completed-word log.
  - Audio toggle (letter sounds on/off, speech recognition prompts on/off).
- 

## 3. Main Features

### 3.1 Letter Sound Playback

- On each letter click, play its phoneme.
- Store short audio files for each letter (mp3 or wav) or use Web Speech Synthesis with per-letter phoneme mappings.

### 3.2 Prefix-Based Letter Filtering

- The app reduces the grid to valid next letters using the prefix map.

### 3.3 Detection of Completed Words

- If prefix matches a word in the dictionary:
  - Apply a visual highlight.
  - Display a “Read this word” button.

### 3.4 Speech Recognition for Word Completion

- Use Web Speech API (when available).
- Speech recognizer listens for a single utterance.
- Compare recognized word to the target prefix.
- Basic tolerance for minor pronunciation variations (normalizing case, stripping spaces).

### 3.5 Success Acknowledgment

- Confetti burst animation overlay on success.
- Spoken or text message such as “Well done”.

### 3.6 Completion Log

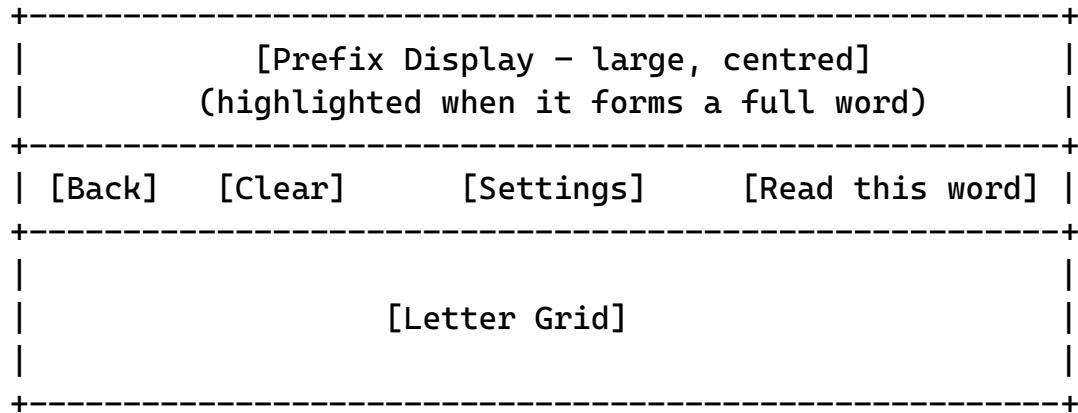
- Maintain a persistent log in localStorage.
- Display log in the settings panel.
- Provide “Clear Log” button.

### 3.7 Word List Management

- Editable via settings.
  - Stored in localStorage.
  - Rebuild prefix map on save.
- 

## 4. UI Specification

### 4.1 Layout Structure



### 4.2 Audio/Recognition Indicators

- Small text or icon when microphone is active.
- Subtle sound or colour cue if recognition fails.

### 4.3 Confetti Animation

- Lightweight canvas animation overlay.
- Runs for ~1–2 seconds.
- Does not obstruct the interface.

### 4.4 Settings Panel

- Word list text area.
- Completed words log (scrollable).
- Toggles:
  - Letter sounds
  - Speech recognition prompts
- Buttons:
  - Save words
  - Clear completed-word log
  - Close settings

---

## 5. Technical Specification

## 5.1 Platform

- Pure front-end: HTML/CSS/JS.
- Must run offline.

## 5.2 Audio Handling

### Letter Sounds

Two options (developer may choose):

1. **Audio files:** /audio/a.mp3, /audio/b.mp3, etc.
2. **Speech Synthesis:** per-letter phoneme mapping.

Trigger:

```
playLetterSound(letter);
```

### Word Spoken Feedback

Use Speech Synthesis for optional spoken encouragement.

## 5.3 Speech Recognition

- Use Web Speech Recognition API where supported.
- Fallback: disable feature gracefully.

Procedure:

1. On “Read this word”, start recognition.
2. On result:
  - Normalize recognized string: lowercase, strip spaces/punctuation.
  - Compare to current prefix.
3. If match: trigger success.
4. If not: notify mismatch.

## 5.4 Prefix Map

Same as previously:

```
prefixMap[prefix] = Set([...validNextLetters]);
```

## 5.5 Completed Word Log

Stored in localStorage:

```
completedWords = ["cat", "home", "car", ...];
```

When word is successfully spoken:

- Append to log.
- Update settings panel.

## **5.6 File Structure**

/index.html  
/styles.css  
/app.js  
/words.txt  
/audio/...

## **5.7 Performance Requirements**

- Must be responsive on low-spec mobile devices.
  - Prefix map rebuilding must be <20 ms for typical word lists.
  - Confetti animation must not cause frame drops.
- 

## **6. Interaction Behaviour**

### **6.1 Letter Selection**

- Append letter to prefix.
- Play sound.
- Animate letters out/in based on new allowed set.

### **6.2 Completing a Word**

- Highlight prefix.
- Show “Read this word” button.

### **6.3 Reading Aloud**

- User taps button.
- Microphone activates.
- Recognition result → success/failure pathway.

### **6.4 Success Path**

- Play confetti.
- Add word to completion log.
- Prompt for new word:
  - “Start new word?” → Clear grid and prefix.

### **6.5 Back and Clear**

- Back removes last character.
  - Clear resets everything.
-

## **7. Developer Deliverables**

- Complete front-end implementation.
- Phonetic letter sounds.
- Working speech recognition and success detection.
- Confetti celebration.
- Settings panel including word-list and log editing.
- Persistence via localStorage.
- Clean, documented code.