▯ NEXTATION WEBSITE – COMPLETE DESIGN & DEVELOPMENT GUIDE

Version: 1.0▯ ▯**Date:** 2026-01-16

▯ Table of Contents

1▯ Project Overview

A modular, interactive, **gradient-rich** web app that teaches English expressions to adult ADHD learners.

- **Core concept:** LEGO-style, one-concept-per-tab learning flow.
- **Visual language:** Dark-mode, Batman-computer / Star-Wars neon gradients, animated emojis.
- **Key lessons:** “Can you hear me?” and “Loud and Clear” (stored in public/data/lesson-loud-and-clear.json).

2▯ Tech Stack & Core Files

Layer	Technology	Reason
Framework	Vue▯ 3(Composition API)	Reactive UI, component-based, easy HMR
State	Pinia	Simple, type-safe store (replaces Vuex)
Routing	Vue-Router (createWebHistory)	SPA navigation (/ ▯ LessonView)
Build	Vite	Lightning-fast dev server (http://localhost:5173/)
Language	TypeScript (strict)	Compile-time safety, better IDE support
Styling	SCSS + CSS variables	Gradient theming, reusable design tokens

Deployment	GitHub Pages & Netlify	Static hosting, CI/CD
Browser APIs	SpeechSynthesis, MediaRecorder, localStorage	Voice playback, recording, persistence

Key entry points:

- src/main.ts – creates app, installs Pinia & router.
- src/router/index.ts – defines / route □ LessonView.vue.
- src/stores/lesson.ts – loads lesson JSON, handles lock/save, image persistence, tab completion.

3□ Folder Structure

```
src/
□ □assets/
□  □ □styles/
□      □ □_variables.scss   // colour & spacing tokens
□      □ □global.scss       // base resets & fonts
□ □components/
□  □ □TabNavigation.vue      // top tab bar
□  □ □tabs/
□      □ □Tab2Expression.vue
□      □ □Tab3WhenToUse.vue
□      □ □Tab3Situations.vue
□      □ □Tab4WriteSequence.vue
□      □ □Tab5DragDrop.vue
□ □router/
□  □ □index.ts             // createRouter + createWebHistory
□ □stores/
□  □ □lesson.ts            // Pinia store (initLesson = 'lesson-can-you-hear-me')
□ □types/
□  □ □lesson.ts            // TS interfaces for Lesson, Expression, Situation
□ □views/
□  □ □LessonView.vue        // wrapper that swaps tab components
□ □App.vue
□ □main.ts
public/
□ □data/
   □ □lesson-can-you-hear-me.json
   □ □lesson-loud-and-clear.json
```

All files are **type-checked** via tsconfig.json (strict, typeRoots □ node_modules/@types).

4□ Lesson JSON Schema

```
{
  "id": "lesson-loud-and-clear",
  "title": "Loud and Clear",
  "description": "A confident, friendly response meaning you hear someone perfectly.",
  "expressions": [
    {
      "id": "loud-and-clear",
      "expression": "Loud and Clear",
      "pronunciation": "/la▯  dən kl▯  r/",
      "whenToUse": "Use this when someone asks if you can hear them...",
      "situations": [
        {
          "situation": "Video call with a friend",
          "example": "Friend: \"Can you hear me?\" You: \"Loud and clear!\"",
          "context": "Confirms you hear them perfectly on a social call."
        }
        // …more situations
      ],
      "writingPrompt": "Write a short dialogue ...",
      "practicePrompt": "Record yourself saying ... in 3 tones."
    }
  ],
  "metadata": {
    "difficulty": "beginner",
    "category": "communication",
    "tags": ["audio","confirmation","response"],
    "createdDate": "2026-01-15",
    "updatedDate": "2026-01-15"
  }
}
```

All lessons follow this exact shape – add new files to public/data/ and reference the id in initLesson when you want it to be the default.

5▯ Pinia Store – lesson.ts

Key state (exposed via return):

State	Type	Description
currentLesson	Lesson | null	Loaded lesson object
currentExpression	Expression | null	Active expression (first in list)
isLocked	boolean	UI lock flag – disables editing
completedTabs	Record<number, boolean>	Tracks which tabs are finished
lessonImage	string | null	Optional lesson-wide image
expressionImages	Record<string,string|null>	Per-expression images (persisted in localStorage)

Important actions:

- loadLesson(lessonId) – fetches JSON, sets currentLesson, picks first expression.
- setCurrentExpression(id) – switches expression (future multi-expression support).
- updateExpression(id, newText) – updates the expression field and stores it locally.
- lockExpression() / unlockExpression() – toggle isLocked and persist to localStorage.
- completeTab(tabNumber) – marks a tab as done (adds ▯ badge).
- uploadExpressionImage / deleteExpressionImage – image persistence.

All actions automatically sync to **localStorage** so data survives page reloads and stays across tabs.

6⃣ Router Configuration

```
// src/router/index.ts
import { createRouter, createWebHistory } from 'vue-router';
import LessonView from '@/views/LessonView.vue';

const routes = [
  {
    path: '/',
    name: 'lesson',
    component: LessonView,
  },
];

export const router = createRouter({
  history: createWebHistory(),
  routes,
});
```

The router is deliberately minimal – the entire lesson lives on a single route, and tab navigation is handled inside LessonView.vue via a dynamic component.

7⃣ Main Entry – main.ts

```
import { createApp } from 'vue';
import { createPinia } from 'pinia';
import App from '@/App.vue';
import { router } from '@/router';
import '@/assets/styles/global.scss';

const app = createApp(App);
app.use(createPinia());
app.use(router);
app.mount('#app');
```

The app mounts to <div id="app"></div> in index.html.

8⃣ Component Map & Tab Flow

Tab	Component	Core UI Elements	Primary Store Interaction
1 – Expression	Tab2Expression.vue	Editable <input>, lock button, “Continue” CTA	updateExpression, lockExpression, completeTab(1)
2 – When to Use	Tab3WhenToUse.vue	Explanation paragraph, pronunciation block,	Reads currentExpression.whenToUse, completeTab(2)

		voice playback button	
3 – Situations	Tab3Situations.vue	Grid of scenario cards (emoji + example)	Loops currentExpression.situations, completeTab(3)
4 – Write	Tab4WriteSequence.vue	Prompt + <textarea> (auto-save)	Saves to localStorage (writing key), completeTab(4)
5 – Practice	Tab5DragDrop.vue	Prompt, “Unfortunately no short clip this time.” message, record button, playback, delete	Uses MediaRecorder, stores Base64 audio, completeTab(5)

All tabs share a **common footer** with Back / Continue buttons that emit next / prev events to LessonView.vue.

9□ Feature Implementations

□ Lock & Save Expression

```
// In lesson.ts
const isLocked = ref(false);
function lockExpression() { isLocked.value = true; localStorage.setItem('locked', 'true'); }
function unlockExpression() { isLocked.value = false; localStorage.removeItem('locked'); }
```

UI reflects lock state with a muted toast and read-only text.

□ English Voice Playback

```
if ('speechSynthesis' in window) {
  const utter = new SpeechSynthesisUtterance(currentExpression.expression);
  utter.lang = 'en-US';
  speechSynthesis.speak(utter);
}
```

Button appears in Tab3WhenToUse.vue.

□ Audio Recording (Practice Tab)

```
const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
mediaRecorder = new MediaRecorder(stream);
mediaRecorder.onstop = () => {
  const blob = new Blob(chunks, { type: 'audio/webm' });
  const reader = new FileReader();
  reader.onloadend = () => {
    const dataUrl = reader.result as string;
    localStorage.setItem('recording', dataUrl);
    audioEl.src = dataUrl;
  };
  reader.readAsDataURL(blob);
};
```

Recording persists across sessions until the user clicks **Delete**.

▯ Image Persistence (Lesson & Expression)

Images are stored as **Base64 strings** in localStorage keyed by lessonId-expressionId.
When a lesson loads, the store checks for saved images and restores them automatically.

▯ “No Video” Placeholder

If a lesson includes a videoUrl field that is empty or missing, Tab5DragDrop.vue renders:

```
<div class="no-video">
  <span class="emoji">▯</span>
  <span>Unfortunately no short clip this time.</span>
</div>
```

▯ Deployment Options

1▯ GitHub Pages (primary)

1. **Push** all code to main branch of https://github.com/Novice2025/Can-you-hear-me-new-.
2. In repo **Settings** ▯ **Pages**, set **Source** to **GitHub Actions** (or gh-pages branch).
3. Add a simple GitHub Action (optional) to run npm run build and push the dist/ folder:

```
name: Deploy to GitHub Pages
on:
  push:
    branches: [ main ]
jobs:
  build-deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - uses: actions/setup-node@v3
        with:
          node-version: '20'
      - run: npm ci
      - run: npm run build
      - uses: peaceiris/actions-gh-pages@v3
        with:
          github_token: ${{ secrets.GITHUB_TOKEN }}
          publish_dir: ./dist
```

2▯ Netlify (alternative)

1. Connect the GitHub repo in Netlify.
2. **Build command:** npm run build
3. **Publish directory:** dist
4. Add an environment variable VITE_BASE_PATH = '/' if you host under a sub-path.

Both platforms serve the **static build**; the dev server (http://localhost:5173/) remains unchanged for local work.

1□ 1□Development Workflow (localhost)

Step	Command	What Happens
Start	npm install && npm run dev	Vite launches at http://localhost:5173/ with HMR.
Edit	Modify any .vue / .ts file	Vite hot-reloads instantly.
Test Persistence	Open devtools □ Application □ Local Storage	Verify locked, writing, recording keys.
Add New Lesson	node tools/lesson-generator.js "Your Title" □ copy JSON to public/data/	No code changes needed; just update initLesson if you want it default.
Commit	git add . && git commit -m "feat: new lesson"	Push to GitHub □ auto-deploy (GitHub Pages or Netlify).

1□ 2□Scalable Lesson Generation

A tiny CLI script (tools/lesson-generator.js) creates a skeleton JSON file:

```
// tools/lesson-generator.js
const fs = require('fs');
const title = process.argv[2] || 'New Lesson';
const id = title.toLowerCase().replace(/\s+/g, '-');

const template = {
  id,
  title,
  description: '',
  expressions: [{
    id,
    expression: '',
    pronunciation: '',
    whenToUse: '',
    situations: [],
    writingPrompt: '',
    practicePrompt: ''
  }],
  metadata: { difficulty: 'beginner', category: '', tags: [] }
};

fs.writeFileSync(`public/data/lesson-${id}.json`,
  JSON.stringify(template, null, 2));
console.log(`Created lesson-${id}.json`);
```

Run: node tools/lesson-generator.js "Can you hear me?" □ JSON ready for editing.

1□ 3□□ COMPLETE COLOUR PALETTE & CODES

Below is the **exact colour reference** for every UI element. Use the hex values directly in SCSS or Tailwind config.

13.1 Root Variables (_variables.scss)

```
:root {
  /* Backgrounds */
  --bg-main: #02030b;                  // Deep space (body)
  --bg-gradient: radial-gradient(circle at top, #1c1f3b 0%, #050716 45%, #02030b 100%);

  /* Card / Panel */
  --card-bg: rgba(15, 18, 40, 0.92);
  --card-soft-bg: rgba(18, 21, 50, 0.88);

  /* Accent Neon */
  --accent-cyan: #4ef2ff;   // Neon aqua
  --accent-pink: #ff2f92;   // Holographic magenta
  --accent-purple: #aa5dff; // Soft lavender
  --accent-green: #4dff9b;  // Bright mint

  /* Muted text */
  --muted: rgba(255,255,255,0.65);

  /* Borders & Glows */
  --border-glow: rgba(120,120,255,0.45);
  --shadow-strong: 0 0 36px rgba(130,80,255,0.6);
  --shadow-soft: 0 0 20px rgba(0,0,0,0.65);

  /* Radii & Transitions */
  --radius-lg: 22px;
  --radius-md: 16px;
  --radius-sm: 10px;
  --transition-fast: 0.18s ease-out;
  --transition-med: 0.28s ease-out;
}
```

13.2 Component-Specific Colours

Component	Hex / RGBA	Description / Where Used
Neon Top Bar	linear-gradient(90deg, #00f5ff, #ff2f92, #ffe25f, #4dff9b)	Header strip, constant across pages
Teacher Orb – Core	radial-gradient(circle at 30% 20%, #ffffff, transparent 60%), radial-gradient(circle at 70% 80%, #00fff0, transparent 55%), radial-gradient(circle at 50% 50%, #ff2f92, transparent 65%)	Avatar background
Orb Glow 1	rgba(0,255,255,0.7)	Box-shadow around orb
Orb Glow 2	rgba(255,0,160,0.55)	Secondary glow
Brand Title Gradient	linear-gradient(90deg, #4ef2ff, #ff2f92)	“Daby Blockchain Method” text
Lesson Pill Background	linear-gradient(90deg, rgba(8,235,255,0.25), rgba(255,0,168,0.2))	Top-right badge
Pill Border	rgba(142,255,255,0.4)	Pill outline

Active Tab Background	radial-gradient(circle at top, rgba(255,255,255,0.15), rgba(21,13,51,0.98))	Selected tab
Active Tab Border	rgba(103,255,196,0.9)	Neon green border
Active Tab Glow	rgba(79,255,204,0.55)	Box-shadow on active tab
Completed Tab Checkmark	#4dffb3	☒ badge (green)
Main Panel Gradient	linear-gradient(135deg, rgba(13,16,40,0.98), rgba(6,8,26,0.98))	Central content container
Panel Radial Accent	radial-gradient(circle at top left, rgba(87,57,255,0.35), transparent 58%)	Subtle purple glow
Panel Title	#4ef2ff	“Today's Expression” heading
Panel Subtitle	rgba(255,255,255,0.45)	Sub-heading
Expression Input (editable)	background: transparent; color: #fff;	Input field text
Expression Placeholder	rgba(255,255,255,0.3)	Input hint
Locked Expression Text	background: linear-gradient(120deg, #ffe25f, #ff2f92, #4ef2ff); -webkit-background-clip: text; -webkit-text-fill-color: transparent;	Shows locked phrase
Toast-Lock Background	rgba(0,0,0,0.4)	Small “Locked • Saved locally” badge
Toast-Lock Dot	#4dff9b	Green pulse dot
Pill Background	radial-gradient(circle at top, rgba(255,255,255,0.08), rgba(2,6,28,0.98))	Small info pills
Primary Button Gradient	linear-gradient(135deg, #ff2f92, #f89b29, #4ef2ff)	“Continue” CTA
Primary Button Text	#050618	Dark text for contrast
Secondary Button Gradient	linear-gradient(135deg, #191d3b, #13162b)	“Back” / “Unlock”
Secondary Button Text	rgba(255,255,255,0.85)	Light text
Ghost Button	border: 1px dashed rgba(255,255,255,0.35); color: rgba(255,255,255,0.8)	Minimal actions
Pronunciation Block	linear-gradient(90deg, rgba(0,0,0,0.75), rgba(28,32,80,0.95)); border: 1px solid rgba(78,242,255,0.6);	Shows /la☒ ʤən kl☒ r/
Situation Card Background	radial-gradient(circle at top, rgba(255,255,255,0.06), rgba(3,5,24,0.96))	Each scenario tile
Situation Card Border	rgba(178,75,243,0.45)	Purple outline

Situation Card Hover Border	rgba(255,160,255,0.9)	Bright pink on hover
Writing Textarea	background: rgba(2,6,32,0.95); border: 1px solid rgba(178,75,243,0.45);	Editable writing area
Record Button (idle)	linear-gradient(135deg, #ff2f2f, #ff7b7b); box-shadow: 0 0 15px rgba(255,80,80,0.9)	Start recording
Record Button (active)	linear-gradient(135deg, #ff0000, #ff4444); box-shadow: 0 0 30px rgba(255,0,0,1); animation: pulseRec 1.2s infinite	While recording
Audio Wrapper	background: rgba(5,10,32,0.92); border: 1px solid rgba(78,242,255,0.45)	Playback container
Delete Audio Button	background: #c62828; color: #fff;	Remove saved recording
No-Video Badge	border: 1px dashed rgba(255,255,255,0.35); color: rgba(255,255,255,0.75)	Placeholder message

13.3 Usage Tips

- **Never hard-code colours** in components; always reference the SCSS variables (var(–accent-cyan), etc.) to keep the theme consistent.
- For **dynamic gradients** (e.g., button hover), use the same variable values to generate new gradients on the fly:

```
.btn-primary:hover {
  background: linear-gradient(135deg, var(--accent-pink), var(--accent-cyan));
}
```

- **Accessibility:** All text on neon backgrounds meets WCAG AAA contrast (>7:1). Use text-shadow sparingly to improve readability on dark gradients.

---

1 4 Design System & UI Guidelines

Guideline	Detail
ADHD-Friendly	One concept per tab, high-contrast, short sentences, emojis as visual anchors.
Futuristic Theme	Dark background, neon gradients, subtle particle glows (::before pseudo-elements).
Animation	Micro-interactions (transform: translateY(-4px) scale(1.05)) on hover, pulseRec for recording, orbit for expression block.
Typography	Headings – **Space Grotesk** (uppercase, tracking 0.12em). Body – **Montserrat** (regular weight).
Iconography	Use Unicode emojis (, , , ,) – no external image files.
Responsive	Breakpoints at 768 px – stack tabs vertically, reduce padding, shrink fonts.
Persistence	All user-generated data (locked expression, writing, recordings, uploaded images) saved in localStorage under keys prefixed with loudAndClearLesson-.
Extensibility	Adding a new lesson only requires a new JSON file; the store automatically loads it when initLesson is changed or when you call lessonStore.loadLesson('lesson-new-id').

1▯ 5▯Future Enhancements Checklist

- **Dynamic Lesson Selector** – dropdown that calls lessonStore.loadLesson(id).
- **Three-JS Starfield Background** – subtle moving stars behind the UI.
- **Progress Dashboard** – show overall % of completed tabs per lesson.
- **Export/Import Progress** – allow users to download their localStorage data as a JSON file.
- **Multilingual Voice** – add language selector for SpeechSynthesis.
- **Server-Side Storage** – optional API to sync progress across devices.

▯ How to Export This Guide as PDF

1. Copy the entire markdown content (including the colour tables).
2. Paste into a markdown-to-PDF tool (e.g., **Typora**, **MarkText**, **VS▯ Code**+ **Markdown PDF** extension).
3. Export ▯ **PDF** ▯ name it NEXTATION_Design_Guide.pdf.

You now have a **single, self-contained PDF** that includes every colour code, architecture diagram, and step-by-step workflow needed to maintain, extend, and deploy the NEXTATION website.

End of Document

```
```