CLAREO is a sophisticated web-based text analysis tool designed to detect potential AI-generated content or plagiarism by analyzing writing style patterns and consistency. It's built as a **single-page React application (SPA)** using modern web technologies.

Technology Stack

- **Frontend Framework:** React 19.2.0 with functional components and hooks
- **Build Tool:** Vite 7.2.4 (fast development server and optimized production builds)
- **Language:** JavaScript (JSX)
- **Deployment:** GitHub Pages (configured with base path /clareo/)
- **Styling:** Custom CSS with CSS variables for theming

Core Architecture (Two Main Functional Modes)

1. **Analysis Mode (src/App.jsx:359-444)**
    a. Single text analysis with detailed linguistic metrics
    b. Provides comprehensive writing quality and pattern detection
2. **Comparison Mode (src/App.jsx:604-718)**
    a. Baseline profile building from multiple student writing samples
    b. Statistical comparison of new work against established patterns
    c. Detects style deviations that may indicate external assistance or AI generation

How The Analysis Works – Text Analysis Pipeline (src/utils/textAnalysis.js)

- When text is submitted for analysis, it goes through multiple analytical layers:

1. **Readability Analysis (textAnalysis.js:54-64)**
    a. **Flesch–Kincaid Grade Level:** Calculates reading difficulty
    b. **Flesch Reading Ease Score:** 0–100 scale of text complexity
    c. Uses syllable counting and sentence length metics
2. **Sentence Analysis (textAnalysis.js: 24-38)**
    a. Breaks text into individual sentences
    b. Counts words and syllables per sentence
    c. Creates detailed sentence profiles
3. **Vocabulary Analysis (textAnalysis.js:66-84)**
    a. **Total Words & Unique Words:** Raw vocabulary metrics
    b. **Type–Token Ratio (TTR):** Basic vocabulary diversity
    c. **MSTTR (Mean-Segmental TTR):** More reliable vocabulary variety measure (calculates TTR in 50-word segments)
    d. **Sophistication Ratio:** Percentage of words ≥6 letters that aren't common
4. **Sentence Variation (textAnalysis.js:281-289)**
    a. **Coefficient of Variation (CV):** Measures natural rhythm vs. algorithmic uniformity
    b. Low CV (<25%) suggests robotic/AI-generated patterns
    c. Calculates standard deviation of sentence lengths relative to mean
5. **Formulaic Language Detection (textAnalysis.js:104-138)**

a. Scans for ~200+ stock phrases from constants/phrases.js
b. Each phrase has a severity weight (1-3)
c. Calculates weighted score indicating AI-like patterns
d. Examples: "in conclusion," "it is important to note," "in today's society"

6. **N-gram Analysis (textAnalysis.js:141-207)**
    a. Detects common bigrams (2-word sequences) and trigrams (3-word sequences)
    b. Compares against human baseline rates
    c. **Predictability Score:** Elevated scores indicate template-driven writing
    d. Uses Map data structure for O(1) lookup performance
7. **Paragraph Flow Analysis (textAnalysis.js:209-279)**
    a. Counts paragraphs and analyzes structure
    b. Coherence Score: Measures topic continuity via shared content words
    c. **Transition Rate:** Percentage of paragraphs starting with connectives
    d. Tracks opening words and sentence distribution
8. **Passive Voice Detection (textAnalysis.js:291-298)**
    a. Regex pattern matching for passive constructions
    b. Example: "was written," "has been done"
    c. Calculates ratio per sentence
9. **Connective Analysis (textAnalysis.js:86-102)**
    a. Categorizes logical connectors (addition, contrast, cause-effect, etc.)
    b. Measures cohesion and argument structure

Statistical Analysis Engine (src/utils/statisticalAnalysis.js)

- The comparison mode uses rigorous statistical methods:
    - **Z-Score Analysis (statisticalAnalysis.js:58-61)**
        - Measures how many standard deviations a value is from the baseline mean
        - Z-scores > 1.5 indicate notable deviations
        - Z-scores > 2.0 are highly significant
    - **Outlier Detection (statisticalAnalysis.js:85-119)**
        - Uses IQR (Interquartile Range) or Z-score methods
        - Identifies and flags anomalous baseline samples
    - **Statistical Significance Testing (statisticalAnalysis.js:158-184)**
        - Determines if differences are meaningful or random variation
        - Three significance levels: low, medium, high

Advanced Fingerprinting Techniques

- **Vocabulary Fingerprinting (src/utils/vocabularyFingerprint.js)**
    - **Signature Words (vocabularyFingerprint.js:85-107)**
        - Uses TF-IDF-inspired scoring to identify characteristic vocabulary
        - Filters out common words and focuses on content words

        - Top 20 signature words create a unique writing "fingerprint"
    - **Vocabulary Entropy (vocabularyFingerprint.js:115-124)**
        - Shannon entropy calculation measures vocabulary diversity
        - Higher entropy = more varied word choice
        - Formula: $-\Sigma(p(word) \times \log_2(p(word)))$
    - **Overlap Scoring (vocabularyFingerprint.js:182-230)**
        - Compares signature word sets between baseline and current text
        - Low overlap (<40%) triggers "Vocabulary Shift" flag

Syntactic Analysis (src/utils/syntacticAnalysis.js)

- Analyzes sentence structure patterns
- Creates syntactic profiles based on clause patterns
- Detects shifts in writing complexity and structure

Error Detection (src/utils/errorDetection.js)

- Identifies common grammatical/spelling patterns
- **Suspiciously Clean Detection:** Flags text with abnormally low error rates compared to baseline
- Particularly useful for detecting AI polish

Composite Scoring System

- **The Style Consistency Score (0-100)** is calculated using weighted components (App.jsx:185-217):
    - Score = (35% × Metric Deviations) + (25% × Vocabulary Overlap) + (20% × Syntactic Patterns) + (15% × Error Consistency) - Special Penalties
    - Score Interpretation:
        - 90-100: Highly Consistent (green)
        - 70-89: Generally Consistent (blue)
        - 50-69: Noticeable Deviation (yellow)
        - 30-49: Significant Deviation (orange)
        - 0-29: Dramatic Change (red)

Style Change Flags (src/constants/thresholds.js:69-100)

- The system generates severity-labeled warnings:
    - **High Severity:**
        - Suspiciously Clean (error rate drops dramatically)
        - Multiple Deviations (3+ metrics simultaneously off)
    - **Medium Severity:**
        - Vocabulary Shift (overlap <40%)

        - Syntactic Shift (structure deviation ≥2.0)
        - Sophistication Jump (Z-score ≥1.8)
    - **Low Severity:**
        - Formulaic Increase (Z-score ≥1.5)

User Interface Components

- **MetricCard (src/components/MetricCard.jsx)**
    - Displays individual metrics with optional warnings
    - Tooltips provide context
- **TextHighlighter (src/components/TextHighlighter.jsx)**
    - Color-codes detected patterns in original text
    - **Modes:** formulaic phrases, n-grams, passive voice, connectives

Interactive Features

- **Keyboard Shortcuts:** Cmd/Ctrl+Enter to analyze
- **Real-time Word Count:** Updates as you type
- **Profile Quality Indicator:** Weak/Good/Strong based on baseline sample count
- **PDF Export:** Generates reports using src/utils/pdfGenerator.js

Data Flow

1. User inputs text → State stored in analysisText or comparisonText
2. Click "Run Analysis" → Triggers runFullAnalysis() function
3. Text is processed through all analysis modules
4. Results stored in component state (results or comparisonResult)
5. UI renders metrics, charts, and highlighted text
6. Comparison mode builds profile from multiple samples using statistical aggregation
7. Current text compared against baseline using z-scores and deviation analysis
8. Flags generated based on threshold crossings
9. Composite score calculated and displayed

Performance Optimizations

- Pre-compiled regex patterns for faster matching
- Map data structures for O(1) n-gram lookups
- useMemo hooks prevent unnecessary recalculations
- useCallback hooks memoize event handlers

Academic Rigor

- The methodology is grounded in established research:
    - Flesch-Kincaid readability (1948/1975)
    - Coh-Metrix framework (Graesser et al., 2004)

    - Statistical significance testing (z-scores, standard deviations)
    - Type-Token Ratio variations for lexical diversity

Deployment Architecture

- Build: npm run build creates optimized production bundle
- Deploy: npm run deploy pushes to GitHub Pages via gh-pages package
- Base Path: Configured for subdirectory hosting at /clareo/
- 404 Handling: index.html includes redirect script for SPA routing