Project Report: My Editor

Project Title

My Editor: A User-Friendly Text Editor for Enhanced Readability Analysis

---

Developer Information

- **Developer:** Adeel Ahmed Qureshi
- **Date of Report:** [22/12/2024]

Abstract

MyEditor is a modern, web-based text editor inspired by Hemingway Editor. The goal of MyEditor is to provide a clean and intuitive platform for users to improve their writing by analyzing readability, tracking word and character counts, and calculating grade levels using the Flesch-Kincaid formula. With features such as real-time updates, dark mode, and responsive design, MyEditor focuses on accessibility and functionality while maintaining a minimalist user experience.

Table of Contents

---

1. Introduction

Writing plays a crucial role in communication, and tools that enhance clarity and readability are invaluable. MyEditor is developed to address this need by providing a platform that analyzes text

readability and assists users in creating concise, easy-to-read content. Unlike advanced editing tools, MyEditor focuses on essential functionalities, offering a lightweight yet powerful tool for writers, students, and professionals.

---

2. Objectives

The primary objectives of MyEditor are:

- To create a text editor that provides real-time readability analysis.
- To calculate readability grade levels using the Flesch-Kincaid formula.
- To offer a user-friendly interface with minimal distractions.
- To support light and dark themes for comfortable use in different environments.
- To ensure responsiveness across devices, including desktops, tablets, and mobile phones.

---

3. Features

3.1 Core Functionalities

- **Content Editing:** A content-editable div for users to write and edit text directly.
- **Real-Time Statistics:** Display word count, character count, sentence count, and readability grade level.
- **Grade Level Calculation:** Leverages the Flesch-Kincaid formula to evaluate the readability grade level.
- **Reset Button:** Clears the editor content and statistics.
- **Theme Toggle:** Allows switching between light and dark modes.

3.2 Advanced Functionalities

- **Responsive Design:** Ensures optimal user experience on devices of all sizes.
- **Routing:** Uses React Router for navigation between pages like Features, About, and Contact without losing user data.
- **Persistent State:** Text input remains intact when navigating between pages.

3.3 Pages

- **Home Page:** Main editor interface.
- **Features Page:** Lists the key functionalities of MyEditor.
- **About Page:** Provides information about the project and its objectives.
- **Contact Page:** Displays a form to get in touch with the developer.

---

4. Design and Architecture

4.1 Frontend

- **Frontend Library:** React (with Vite for fast development and optimized builds).
- **Styling:** CSS modules for global styles and component-specific styling.
- **Libraries:**
  - `syllable` for counting syllables in words.
  - `react-router-dom` for client-side routing.

4.2 Components

- **Navbar:** Provides navigation links to different pages with theme toggle functionality.
- **Editor:** The main content-editable area where users can write and edit text.
- **Sidebar:** Displays real-time readability statistics and utility buttons.
- **Features/About/Contact:** Static components displaying project-related information.

4.3 State Management

- State variables are managed using React hooks (`useState` and `useRef`) for:
  - Text content persistence.
  - Grade level calculation.
  - Theme toggle.

---

5. Implementation Details

5.1 Key Algorithms

Flesch-Kincaid Grade Level Formula:

The formula calculates the readability of text based on sentence length and syllable count:

Word Count:

Counts words using whitespace separation with regex.

Sentence Count:

Matches sentences using regex for punctuation (e.g., `.!?`).

Syllable Count:

Uses the `syllable` library to count syllables in each word.

5.2 Routing

React Router is used for navigation across Home, Features, About, and Contact pages without reloading the app. Text content remains persistent during navigation.

5.3 Responsive Design

CSS media queries ensure that the application adapts to different screen sizes. For example, the layout adjusts from a grid view to a single-column view on smaller devices.

---

6. Challenges Faced

- **State Persistence:** Ensuring the text remains intact when navigating between pages was challenging but resolved by lifting the text state to the App component.
- **Readability Calculations:** Accurately counting syllables, sentences, and grade levels required the use of external libraries and careful testing.
- **Responsive Design:** Maintaining usability and aesthetic appeal across different screen sizes required extensive use of CSS media queries.

---

7. Future Scope

- **Advanced Readability Features:** Highlighting complex sentences and passive voice detection.
- **Export Options:** Allow users to save text as a file (e.g., `.txt` or `.docx`).
- **Multi-Language Support:** Extend readability analysis to other languages.
- **User Authentication:** Add user profiles to save and manage multiple documents.

---

8. Conclusion

MyEditor successfully combines simplicity and functionality to create an effective text editor for readability analysis. The project demonstrates the potential of web-based tools to enhance writing quality and user experience. With further development, MyEditor can evolve into a comprehensive writing assistant for diverse audiences.

Appendix

8.1 Tools and Technologies

- **Frontend:** React
- **Development Environment:** Vite
- **Libraries:** React Router, Syllable
- **Styling:** CSS modules

8.2 References

- Flesch-Kincaid Readability Formula: https://en.wikipedia.org/wiki/Flesch%E2%80%93Kincaid_readability_tests
- Syllable Library: https://github.com/words/syllable