DOCUMENTATION

Reach Radar –Social Media Content Analyzer

Reach Radar is a website that analyzes your social media text content, such as posts, evaluates their tone and sentiments, and provides suggestions to improve engagement and reach.

Tech Stack:

- **Vite**: For creating the web application
- **React**: For building the user interface
- **HTML**: For structuring the web pages
- **Tailwind CSS**: For styling the application
- **JavaScript**: For adding interactivity and functionality

Features:

The site has following Features which are Achieved through following approach:

1. **Drag and Drop Functionality:** In addition to allowing file selection through a button, the site provides a drag-and-drop feature where users can drop files into a designated area. This is implemented by making the div area selectable and handling file drop events.

2. **Handling Types of Files: - The** site only allows files in PDF or image formats (such as JPG or PNG). This is achieved through file validation during file handling.

3. **Extracted Text: - The** extracted text from both file types is displayed in extracted text box. The approach used to extract text is as follows:

    A) **Extracting from Pdf**: - To extract text from PDF files, the library react-pdftoText is used.

    Other Alternatives: - [pdf-lib, pdf.js, pdf-parse]

    B) **Extracting from images: -** for this task react library tesseract.js is used.

Other alternatives [ocrad.js, Google vision API, Aws textract].

4. **Analysis Page: -** The extracted Text is then analyzed, and certain insights are presented as follows: -

   A) **Sentiment: -** The extracted text is sent to a model (used via API) from Hugging Face to analyze its sentiment as **POSITIVE**, **NEGATIVE**, or **NEUTRAL**.
   Other alternative API's- [IBM Watson cloud, NLP cloud, Meaning Cloud]

   B) **Hashtag Suggestion: -** The extracted text is fed into a text-generation model from Hugging Face to analyze the content and extract keywords that can be used as hashtags to increase post visibility. The model processes the input using the following prompt, "Extract **relevant hashtags from the following text to maximize social media engagement. Return only hashtags, separated by spaces. Here is the input text: "${text}"** The model outputs a string of hashtag keywords, separated by spaces, which are then displayed in the output section.
   Other Alternatives API's - [Open AI, Google gemma API, IBM Watson]

   C) **Hashtag Copy: -** A **Copy** button in the Suggested Hashtags section enables users to copy the generated hashtags easily and paste them directly into their **posts**.

   D) **Suggestions: -** The extracted text is fed into a text-generation model to analyze its tone and engagement level. The model then provides actionable suggestions to improve the post's effectiveness and popularity. This functionality uses the Hugging Face Text Generation API with the following prompt, **"Analyze the following social media content and suggest specific improvements to maximize user engagement, reach, and impact. Focus on aspects such as tone, formatting, and content relevance. Exclude any suggestions related to hashtags. Provide the output as a plain string. The first sentence should be the heading like 'You can improve your post in the following ways,' followed by individual points on how to improve the content also there should be no symbol : "${text}".** The output from the model is further processed to improve the formatting and presentation of the text. This ensures the suggestions are clear, concise, and easy to follow. The refined suggestions are then displayed in the Suggestions Section for the user to review and implement.
   Other Alternatives API's - [Open AI, Google gemma API, IBM Watson]

Best Practices

1) **Using Toast for Notifications:** Notifications are displayed using a toast library (e.g., react-toastify) instead of browser alerts to provide a non-intrusive and visually appealing user experience.
2) **Storing API Keys Securely:** API keys are stored in a .env file to enhance security and prevent unauthorized access. This ensures sensitive credentials are not exposed in the codebase.

Challenges

1) **Creating a Responsive UI/UX:** Designing a user interface that adapts seamlessly to different screen sizes and devices while maintaining a visually appealing and user-friendly experience.

2) **Finding APIs with High Free Limits:** Identifying APIs that offer a high number of free requests per day while meeting the project's requirements for functionality and reliability.
3) **Seamless API Integration:** Integrating the selected APIs into the project without disrupting the existing structure, ensuring modularity, and maintaining clean, efficient code.
4) **Pre-processing Text for Desired Format:** Processing the raw output from the text-generation API to refine and present the suggestions or extracted data in a clean, structured, and user-friendly format.