**Tech Stack:**

Python, Javascript, Sql, React native, Fastapi, Spacy, Nltk, Opencv, Tensorflow, Librosa, Scikit-learn, Jwt, Aes, Thunder client, Pandas, Numpy

**📅 Implementation Plan (3 Months)**

**✅ Phase 1: Preparation & Setup (Week 1)**

**Objective:** Establish the foundation for the project and set up development environments.

1. **Environment Setup:**
   * Install necessary libraries and frameworks (FastAPI, React Native, spaCy, NLTK, OpenCV, TensorFlow, librosa, scikit-learn, pandas, numpy).
   * Set up Python virtual environments for backend.
   * Set up React Native environment for frontend (Node.js, npm, Android Studio/Expo).
2. **Project Structure:**
   * Backend (FastAPI): Create project folders for APIs, AI models, authentication, and database interactions.
   * Frontend (React Native): Initialize React Native app with basic routing and UI setup.
   * Database (MySQL): Define the database schema for storing user data, feedback history, and content metadata.
3. **Version Control:**
   * Initialize Git repository.
   * Set up branches for different components (frontend, backend, AI models).

**✅ Phase 2: Backend & AI Model Development (Weeks 2 - 4)**

**Objective:** Build and train AI models and create APIs to expose their functionalities.

**1. Text Analysis Module (Week 2)**

* Implement text analysis using **spaCy & NLTK**.
* Features: Grammar checking, readability assessment, tone analysis.
* API Endpoint: /api/text-analysis (POST).

**2. Image Analysis Module (Week 3)**

* Implement image analysis using **OpenCV & TensorFlow**.
* Features: Object detection, quality assessment, style analysis.
* API Endpoint: /api/image-analysis (POST).

**3. Audio Analysis Module (Week 4)**

* Implement audio analysis using **librosa & scikit-learn**.
* Features: Speech-to-text, audio quality assessment, sentiment analysis.
* API Endpoint: /api/audio-analysis (POST).

**4. Security & Authentication (Week 4)**

* Implement **JWT for user authentication**.
* Implement **AES encryption** for securely storing files.

**✅ Phase 3: Frontend Development (Weeks 5 - 6)**

**Objective:** Build the user interface and connect it to backend APIs.

1. **UI Design & Implementation (Week 5)**
   * Create React Native components for:
     + User login & registration.
     + Uploading text, images, and audio files.
     + Displaying feedback results.
2. **API Integration (Week 6)**
   * Connect frontend to backend using **Axios or Fetch API**.
   * Implement user authentication with JWT.
   * Display feedback results in a user-friendly manner.

**✅ Phase 4: Testing & Optimization (Weeks 7 - 8)**

**Objective:** Ensure smooth functionality and optimize performance.

1. **Testing (Week 7)**
   * Backend: Test APIs using **Thunder Client**.
   * Frontend: Test user interactions and API connections.
   * Unit Testing: Use **PyTest** for backend and manual testing for frontend.
2. **Optimization (Week 8)**
   * Enhance model performance using **ONNX (optional)**.
   * Clean up and organize code.
   * Ensure smooth communication between frontend and backend.

**✅ Phase 5: Finalization & Deployment (Weeks 9 - 12)**

**Objective:** Complete the project and prepare for submission.

1. **Documentation (Week 9 - 10)**
   * Write technical documentation for APIs, AI models, and frontend components.
   * Include instructions for installation, usage, and troubleshooting.
2. **Local Deployment (Week 11)**
   * Deploy the backend locally using FastAPI.
   * Test the complete system end-to-end.
3. **Final Adjustments (Week 12)**
   * Fix any remaining bugs.
   * Review the entire system for usability and performance.

**📌 Milestones Overview:**

* **Week 1:** Setup & Structure
* **Weeks 2-4:** Backend & AI Models
* **Weeks 5-6:** Frontend Development
* **Weeks 7-8:** Testing & Optimization
* **Weeks 9-12:** Documentation, Deployment & Final Adjustments

### ✅ ****Phase 1: Preparation & Setup (Week 1)****

#### **1. Environment Setup:**

* **Python Installation:**
  + Ensure **Python 3.10+** is installed.
  + Install virtual environment tool (pip install virtualenv).
  + Create a virtual environment (python -m venv venv) and activate it.
* **Node.js & npm Installation:**
  + Install **Node.js & npm** for React setup.
  + Verify installations (node -v and npm -v).
* **Python Packages Installation:**

bash

CopyEdit

pip install fastapi uvicorn spacy nltk opencv-python-headless tensorflow librosa scikit-learn pandas numpy pyjwt cryptography

* **React App Setup:**

bash

CopyEdit

npx create-react-app critique-app

cd critique-app

npm install axios

npm install react-router-dom

* **Testing Tools Installation:**
  + Install **Thunder Client (VS Code Extension)** for API testing.

#### **2. Project Structure:**

* **Backend (FastAPI):**
  + Create folders: /api, /models, /database, /security, /utils.
  + Create a FastAPI entry file (main.py).
  + Implement basic endpoints for text, image, and audio processing.
  + Implement authentication & security modules (/security).
  + Implement data processing modules (/api/text, /api/image, /api/audio).
  + Implement database models and connections (/database).
* **Frontend (React):**
  + Create folders: /src/components, /src/services, /src/pages, /src/styles.
  + Set up routing using react-router-dom.
  + Implement components for **Login, File Upload, Analysis Results, and History Display**.
  + Connect frontend with FastAPI backend using Axios (/src/services).
  + Set up JWT handling for authentication.

#### **3. Database (MySQL):**

* Install and set up **MySQL Server**.
* Define schema for storing:
  + Users (ID, Username, Password Hash, Token).
  + Feedback history (ID, UserID, ContentType, Result, Timestamp).
* Test database connection with FastAPI.

#### **4. Version Control:**

* Initialize **Git repository** (git init).
* Create a .gitignore file for Python and React.
* Push code to a remote repository (GitHub/GitLab).
* Create separate branches:
  + main (Stable code)
  + backend (FastAPI development)
  + frontend (React development)

#### **5. Initial Testing:**

* Test API endpoints using Thunder Client.
* Test React app connectivity with FastAPI.
* Test JWT authentication.

**🔍 Now, the next step is to start building the actual backend functionalities as listed in your checklist:**

* **Text Analysis Module**
* **Image Analysis Module**
* **Audio Analysis Module**
* **Authentication & Security (JWT handling)**
* **Database Integration (User & Feedback storage)**

1. **Dynamic Improvement Suggestions:**
   * Provide specific suggestions for readability, clarity, grammar, etc., rather than generic feedback.
2. **Spelling Check:**
   * Implement a spelling checker (e.g., textblob or pyspellchecker) to identify and suggest corrections for misspelled words.
3. **Keyword Extraction:**
   * Extract important keywords from the text for enhanced feedback and analysis.
4. **Extended Grammar Analysis:**
   * Enhance grammar checks by flagging incomplete sentences, awkward phrasing, and other common errors, not just case errors.
5. **Detailed Readability Tips:**
   * Provide suggestions for improving readability (e.g., simplifying complex words, breaking down long sentences).
6. **Clarity Tips:**
   * Suggest replacing filler words with more precise words, reducing redundancy, etc.
7. **Tone Analysis:**
   * Detect the current tone of the text (e.g., formal, casual, professional, persuasive, etc.).
   * Provide suggestions if the tone is inappropriate or inconsistent.
8. **Tone Transformation:**
   * Allow the user to specify a desired tone and modify the text to match the requested tone.
9. **Format Adaptation:**
   * Provide structured suggestions to adapt text to a specified format (e.g., formal writing, casual conversation, email style, etc.).
10. **Advanced Sentiment Analysis:**

* Detect nuanced emotions (e.g., Joy, Sadness, Anger, Fear, etc.) beyond just Positive, Negative, or Neutral.
* Provide suggestions to adjust the sentiment if requested.

1. **Combined Analysis & Suggestions:**

* Integrate all aspects (Grammar, Readability, Clarity, Spelling, Tone, Sentiment, Format) into cohesive feedback.

1. **Improvement Suggestions:**

* Provide specific tips on how to enhance clarity, readability, and tone based on analysis results.

#### 1. **Grammar & Punctuation:**

* Subject-Verb Agreement
* Sentence Fragments
* Run-On Sentences
* Misplaced Modifiers
* Punctuation Errors (commas, semicolons, quotation marks, etc.)
* Verb Form & Tense Issues
* Incorrect Word Usage (e.g., affect vs. effect)

#### 2. **Spelling:**

* Misspelled Words
* Contextual Spelling Errors (e.g., their vs. there vs. they’re)

#### 3. **Clarity:**

* Wordiness & Redundancy
* Ambiguous Phrasing
* Overly Complex Sentences (suggests breaking them down)
* Passive Voice (and suggests converting to active voice if needed)
* Overuse of certain words or phrases

#### 4. **Readability:**

* Readability Score (similar to Flesch Reading Ease)
* Sentence Structure Analysis (suggests simplification if too complex)
* Readability Style Suggestions (making sentences smoother and clearer)

#### 5. **Tone Detection & Adjustment:**

* Detects tones like Formal, Friendly, Confident, Enthusiastic, etc.
* Provides suggestions to change the tone to match the user's intended style.

#### 6. **Style & Consistency:**

* Consistency in punctuation, formatting, etc.
* Detection of informal language in formal writing.
* Consistent use of tense and perspective.

#### 7. **Engagement:**

* Suggests more engaging vocabulary and sentence structures.
* Recommends stronger word choices for impact.

#### 8. **Delivery Suggestions:**

* Provides suggestions for adjusting tone based on the user's goals (e.g., making text sound friendly, confident, persuasive, etc.)
* Adjusts formality level based on the intended audience.

#### 9. **Plagiarism Checking (Premium Feature):**

* Checks the text against a huge database of online sources.

**1. Grammar, Punctuation, & Spelling Analysis (Basic Error Detection & Correction)**

* Detect grammar errors (e.g., subject-verb agreement, tense issues, misplaced modifiers, etc.)
* Detect punctuation errors (e.g., missing commas, incorrect punctuation usage, etc.)
* Spell checking & correction using libraries like pyspellchecker or textblob
* Provide detailed suggestions for correcting identified errors.

**2. Readability Analysis & Improvement Suggestions**

* Calculate readability scores (e.g., Flesch Reading Ease, Gunning Fog, etc.)
* Provide detailed suggestions for improving readability:
  + Simplify complex words
  + Break down long sentences
  + Suggest shorter paragraphs
  + Avoid jargon and overly technical language unless required.

**3. Clarity Analysis & Improvement Suggestions**

* Detect unclear sentences, awkward phrasing, and redundancy.
* Suggest better phrasing and concise alternatives.
* Identify filler words and recommend their removal or replacement.

**4. Tone Analysis & Transformation**

* Detect the current tone of the text (e.g., formal, casual, professional, friendly, persuasive, assertive, etc.)
* Provide suggestions if the tone is inconsistent or inappropriate for the intended purpose.
* Allow users to select a desired tone and provide tailored suggestions to match it.

**5. Engagement & Delivery Analysis**

* Identify whether the writing is engaging (e.g., through sentence variety, avoiding passive voice, etc.)
* Provide suggestions for enhancing engagement (e.g., using active voice, rhetorical questions, etc.)
* Check for appropriate sentence flow and coherence.

**6. Keyword Extraction & Analysis**

* Extract important keywords from the text using libraries like spaCy or RAKE.
* Provide keyword density analysis and suggest enhancements for keyword usage based on user goals.

**7. Sentiment Analysis & Transformation**

* Detect nuanced emotions (Joy, Sadness, Anger, Fear, Surprise, etc.) beyond Positive, Negative, or Neutral.
* Provide suggestions to adjust the sentiment to match user preferences (e.g., making a review sound more positive, making feedback sound more constructive, etc.)

**8. Goal Customization & Tailored Feedback**

* Allow users to specify their goals (e.g., Academic writing, Professional emails, Creative writing, etc.)
* Provide customized feedback based on selected goals:
  + Academic Writing: Focus on clarity, grammar, and formal tone.
  + Professional Writing: Focus on professionalism, correctness, and coherence.
  + Creative Writing: Focus on engagement, tone, and emotional impact.
  + Casual Writing: Focus on friendliness, readability, and conversational style.

**9. Format Adaptation**

* Provide structured suggestions to adapt text to a specific format or style (e.g., converting casual text into formal, adjusting text to suit different platforms like emails, reports, articles, etc.)

**10. Comprehensive Suggestions System**

* Integrate all feedback (Grammar, Punctuation, Readability, Clarity, Tone, Engagement, Sentiment) into cohesive suggestions.
* Ensure suggestions are displayed in a user-friendly way (highlighted text, clickable suggestions, etc.)

**11. Text Rewriting Capability**

* Provide an option for automatic rewriting based on user-selected goals or tone.
* Offer alternative sentences or paragraphs to match desired requirements.

**12. Feedback Presentation**

* Present suggestions and improvements in a clear, organized format.
* Provide explanations for suggestions to help users understand why changes are recommended.

1. grammar(e.g., subject-verb agreement, tense issues, misplaced modifiers, etc.)
2. punctuation(e.g., missing commas, incorrect punctuation usage, etc.)
3. spell check
4. readability score
5. provide detailed suggestions for improving readability (this must be unique and to the context of what the input text is of.... eg. if the input is "Due to the excessive utilization of multifarious terminologies, comprehension is substantially hindered" and the output should be 'The sentence uses unnecessarily complex words: “excessive utilization,” “multifarious terminologies,” “comprehension,” and “substantially hindered.”')
6. provide corrected sentence with improved readability(eg. "Using too many different words makes it hard to understand")
7. detect unclear sentences, awkward phrasing, and redundancy (e.g., "She said he told her that the plan they discussed might be changed if they find something better.")
8. provide suggestion on improving clarity (eg. 'The sentence is too vague and has ambiguous references (e.g., “she,” “he,” “they”).')
9. provide corrected sentence with improved clarity (eg. “Emily said that John mentioned their plan might be changed if the team discovers a better option.”)
10. detect tone of the text(e.g., formal, casual, professional, friendly, persuasive, assertive, etc.)
11. provide suggestion if the tone is inconsistent or inappropriate for the intended purpose
12. take user selection of tone and change the input text in required tone
13. identify whether the writing is engaging
14. provide suggestions to enhance engagement
15. check for appropriate sentence flow and coherence
16. provide keyword density analysis and suggest enhancements for keywords usage based on user goals
17. detect nuanced emotions (Joy, Sadness, Anger, Fear, Surprise, etc.)
18. take user selection of emotion and change the input text in required emotion
19. convert the input text as per user required goal(eg, academic writing: focus on clarity grammar and formal tone; professional email: focus on professionalism, correctness and coherence; creative writing: focus on vocabulary, engagement, tone, and emotional impact; casual writing: focus on friendliness, readability; etc.....)
20. offer alternate words, sentences or paragraphs to increase the overall readability, tone, clarity or any other aspect
21. provide explanation for the changed suggested and made to help user understand why changes were suggested or made

|  |  |
| --- | --- |
| **Requirement** | **Best Offline Solution (Highest Satisfaction)** |
| **1. Grammar Check** | T5 (Pretrained) |
| **2. Punctuation Check** | T5 (Pretrained) |
| **3. Spell Check** | T5 (Pretrained) |
| **4. Readability Score** | textstat (Flesch Reading Ease, Gunning Fog, etc.) |
| **5. Readability Suggestions** | LanguageTool + Custom NLP Rules |
| **6. Corrected Sentence (Readability)** | T5 (Pretrained) |
| **7. Detect Unclear Sentences** | SpaCy Dependency Parsing (Identifies vague references) |
| **8. Clarity Suggestions** | LanguageTool + Custom NLP Rules |
| **9. Corrected Sentence (Clarity)** | T5 (Pretrained) |
| **10. Detect Tone** | BERT-Tone-Classifier (Detects formal, casual, persuasive, etc.) |
| **11. Tone Suggestion** | BERT-Tone-Classifier + Rule-Based Adjustments |
| **12. Rewrite in Specific Tone** | T5 (Fine-Tuned for Style Transfer) |
| **13. Engagement Analysis** | TF-IDF + TextRank (Finds repetitive/dull sections) |
| **14. Engagement Suggestions** | TF-IDF + ParaBank2 (Suggests engaging phrasings) |
| **15. Sentence Flow & Coherence** | TextRank + SpaCy (Analyzes sentence transitions) |
| **16. Keyword Density Analysis** | TF-IDF (NLTK/sklearn) (Analyzes keyword usage) |
| **17. Emotion Detection** | GoEmotions (Google AI) (Detects joy, sadness, anger, etc.) |
| **18. Rewrite for Emotion** | T5 (Fine-Tuned for Emotion-Based Writing) |
| **19. Rewrite Based on User Goal** | T5 (Fine-Tuned for Writing Goals) |
| **20. Alternate Words & Sentences** | T5 (Fine-Tuned for Paraphrasing) |
| **21. Explanation for Changes** | Rule-Based Explanation Engine (Combines T5 output + NLP insights) |

**Multi-step pipeline and data flow**

[Raw Text Input]

↓

[1. Sentence Segmentation & Basic Cleaning]

↓

[2. Grammar & Spelling Correction (using seq2seq model)]

↓

[3. Structural Flow Detection (sentence boundaries, run-ons)]

↓

[4. Clarity & Readability Checks (parse trees + heuristics)]

↓

[5. Keyword Analysis (TF-IDF / RAKE / YAKE)]

↓

[6. Sentiment & Emotion Analysis (BERT-based)]

↓

[7. Tone Detection & Transformation (optional rewriting)]

↓

[8. Aggregated Suggestions + Explanations]

↓

[Final Annotated Output: Corrected Text + Suggestion Blocks]

project/

├── .venv/

├── modules/

│ ├── audio\_analysis/

│ ├── image\_analysis/

│ └── text\_analysis/

│ ├── analyzer.py # Orchestrator for entire text pipeline

│ ├── grammar\_corrector.py # Grammar correction logic

│ ├── sentence\_splitter.py # Chunking (new file)

│ ├── readability\_checker.py # Readability checker

│ ├── tone\_emotion\_analyzer.py # Emotion and tone detection

│ ├── keyword\_density.py # Keyword relevance and density

│ └── eval.py # For isolated testing

**✅ Completion Checkpoints (With Order & Goal)**

| **Checkpoint** | **Task** | **Description** |
| --- | --- | --- |
| ✅ CP-1 | Grammar Correction | Current vennify/t5-base-grammar-correction — integrated |
| 🟩 CP-2 | Sentence Boundary Fixing | Detect run-ons, split or join sentences correctly |
| 🟩 CP-3 | Readability / Clarity Module | Use spaCy or syntax trees to detect awkward, unclear or long sentences |
| 🟨 CP-4 | Keyword Analysis | Integrate RAKE / YAKE to extract important terms and check density |
| 🟨 CP-5 | Sentiment & Emotion Detection | Use cardiffnlp/twitter-roberta-base-sentiment or similar |
| 🟨 CP-6 | Tone Analyzer | Build a custom rule-based or small classifier to detect tone (formal/informal, assertive, etc.) |
| 🟦 CP-7 | Suggestion Engine | Unify outputs from all modules into a single response format with detailed explanations |
| 🟦 CP-8 | Output Formatter & Frontend API | Beautify and deliver results to frontend with JSON grouping by category |

**✅ Implementation Order Breakdown**

| **Order** | **Checkpoint** | **Why It Comes Here** |
| --- | --- | --- |
|  | **CP-1: Grammar Correction** | Already done — this gives you the foundation to clean up basic sentence structure and errors. |
|  | **CP-2: Sentence Boundary Detection / Text Flow Fix** | Run-on sentences, missing periods, and improper joins need to be split correctly **before** any clarity or readability assessment. |
|  | **CP-3: Clarity & Readability** | After sentence flow is fixed, this checks how understandable and readable the content is (complexity, passive voice, long sentences, etc.). |
|  | **CP-4: Keyword Analysis** | Keywords are meaningful only after clarity is established. This adds a layer of **goal relevance** and highlights important parts of content. |
|  | **CP-5: Sentiment & Emotion** | Now that text is clean and meaningful, emotion analysis becomes more accurate and useful. |
|  | **CP-6: Tone Detection / Adjustment** | Tone rides on top of structure + emotion. It needs everything before it to be fixed so that it can understand whether content sounds formal, sarcastic, empathetic, etc. |
|  | **CP-7: Suggestion Engine** | Once all analyses are available, you aggregate them and create a **unified suggestion block** — this is the "magic box" that explains why something was flagged. |
|  | **CP-8: Output Formatter / API** | Last — wrap the results into clean JSON or UI-bound responses for presentation. |