AI-Powered Reading Assistant for Dyslexic Students Using OCR and TTS

21CSC203P/ADVANCED PROGRAMMING PRACTICE

PROJECT REPORT

Submitted by

RUPA PRIYA M (RA2411003020272)

GVSS NITHEESHA (RA2411003020318)

Under the guidance of

Jean Justus

(Associate Professor, Department of Computer Science and Engineering)

in partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Of

FACULTY OF ENGINEERING AND TECHNOLOG

SCHOOL OF COMPUTER SCIENCE ENGINEERING

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY RAMAPURAM, CHENNAI - 600089

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Deemed to be University U/S 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report **AI-Powered Reading Assistant for Dyslexic Students Using OCR and TTS** is the bonafide work of **RUPA PRIYA M (RA2411003020272), GVSS NITHEESHA (RA2411003020318)** who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an occasion on this or any other candidate.

SIGNATURE

Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram, Chennai.

SIGNATURE

Computer Science and Engineering,
SRM Institute of Science and Technology,
Ramapuram, Chennai.

Submitted for the project viva-voce held on_____ at SRM Institute of Science and Technology, Ramapuram, Chennai- 600089.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

RAMAPURAM, CHENNAI - 89

DECLARATION

We hereby declare that the entire work contained in this project report titled **AI-Powered Reading Assistant for Dyslexic Students Using OCR has been carried out by RUPA PRIYA M (RA2411003020272), GVSS NITHEESHA (RA2411003020318)** at SRM Institute of Science and Technology, Ramapuram, Chennai, under the guidance of **Jean Justus Associate professor**, Department of Computer Science and Engineering.

Place: Chennai **RUPA PRIYA M**

Date: **G.V.S.S NITHEESHA**

\

ABSTRACT

This project introduces an **AI-powered reading assistant** designed to specifically aid dyslexic students by integrating **Optical Character Recognition (OCR)** and **Text-to-Speech (TTS)** technologies. The system's primary function is to transform inaccessible visual text, including printed documents, low-quality scans, and common handwritten notes, into an accessible auditory format.

The workflow begins with input acquisition (camera capture or file upload), followed by the OCR module, which utilizes the **Tesseract LSTM engine** for text recognition. This text is then synthesized into natural-sounding speech. The assistant's crucial feature is the **synchronized, word-by-word text highlighting** displayed in the user interface, a scientifically proven method that reinforces the visual-auditory link essential for dyslexic learners' comprehension and fluency.

The application is built as a highly customizable, low-cost solution, offering users control over dyslexia-friendly fonts, reading speed, and voice pitch. By fostering independent reading and reducing reliance on external assistance, this tool aims to boost academic performance and confidence. The project represents a practical contribution to digital accessibility, benefiting not only dyslexic individuals but also language learners and those with visual impairments.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

- AI-Artificial Intelligence
- OCR-Optical Character Recognition
- TTS-Text-to-Speech
- LSTM-Long Short-Term MEMORY
- WPM-Words Per Minute
- CER-Character Error Rate
- DFD-Data Flow Diagram
- NFR-Non-Functional Requirement

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Dyslexia is a common learning disability that fundamentally affects the ability to read and decode written language. This challenge stems from a neurological difference in the way the brain processes phonemes (the basic sound units of language), making the visual-to-phonological translation process slow and error-prone. Consequently, reading becomes an exhaustive task, often diverting cognitive resources away from comprehension and significantly impacting academic engagement.

Traditional solutions, such as human readers or basic screen-reading software, often lack the necessary integration and customization required for effective therapeutic aid. This project recognizes the potential of modern Artificial Intelligence (AI), specifically in the domains of Computer Vision and Natural Language Processing, to create a sophisticated yet accessible intervention tool. By developing an integrated system that captures diverse physical text and instantly translates it into a supportive audio-visual format, we aim to overcome these foundational reading barriers.

1.2 PROBLEM STATEMENT:

The core problem addressed by this project is the lack of a single, **versatile, and therapeutically effective reading assistant** that is both highly accurate and affordable for dyslexic students. Existing gaps include:

1. **Limited Input Flexibility:** Most tools fail when presented with varied student inputs, such as low-quality document scans, complex non-standard fonts, or **handwritten notes**, forcing students to transcribe text manually.
2. **Lack of Bimodal Synchronization:** Simple TTS readers do not offer **synchronized word-by-word highlighting**, which is crucial for training the dyslexic brain to track text visually while hearing the corresponding sound.
3. **High Cost of Customization:** Commercial products offering these features are often expensive, creating a financial barrier to access essential educational support tools.

.

1.3 OBJECTIVES:

The overall Goal is to design, implement, and validate a seamless AI-Powered Reading Assistant that converts physical and digital text inputs into an accessible, synchronized audio-visual reading experience for dyslexic students.

The specific Objectives are:

1. Robust Input Acquisition (FR01-FR05): Create a module capable of handling images, PDFs (using Poppler), DOCX, and live camera capture (using OpenCV), normalizing all inputs for the OCR engine.
2. High-Accuracy OCR Pipeline (FR06): Integrate the Tesseract OCR engine and implement robust image pre-processing (binarization, deskewing) to ensure high text recognition accuracy, especially for moderately challenging inputs.
3. synchronized Bimodal Output (FR07, FR09): Develop a system that generates natural-sounding TTS and couples the audio playback with precise, word-level highlighting in the web interface.

1.4 SCOPE AND MOTIVATION:

Scope

The current project implementation focuses on the core, end-to-end pipeline:

- **Technology Stack:** Python (Flask, OpenCV, Tesseract, pdf2image) and JavaScript/HTML/CSS for the frontend.
- **Input Support:** JPG, PNG, PDF, and DOCX files; raw text paste; and single-frame camera capture.
- **Language:** English language OCR and TTS processing.
- **Customization:** Local persistence of reading speed, pitch, and font selection.

Motivation

Our motivation is to harness open-source AI technologies (Tesseract, OpenCV, Flask) to build a **low-cost, highly customizable, and independent reading solution**. This system will empower students to engage with *any* text source from classroom handouts to textbooks—fostering confidence and reducing their dependence on others.

CHAPTER 2
EXISTING SYSTEM

2.1 Overview of Existing Reading Aids

Existing solutions for reading assistance generally fall into two categories: traditional screen readers and proprietary commercial solutions.

1. **Traditional Screen Readers (e.g., NVDA, macOS VoiceOver):** These are excellent for digital text (TXT, DOCX, Web content) but completely fail when presented with physical documents, image scans, or non-machine-readable text. They lack built-in OCR capabilities.
2. **Basic OCR Tools (e.g., Google Lens, simple apps):** These can convert an image to text, but the output is static. They lack the integration with TTS and, critically, do not provide the essential **synchronized word-by-word highlighting** required for effective bimodal learning and therapeutic aid for dyslexic students.
3. **High-End Commercial Tools (e.g., ClaroRead, Kurzweil 3000):** These offer the necessary bimodal features, advanced OCR, and custom fonts. However, their subscription models and high cost create a significant financial barrier, preventing widespread adoption in low-resource settings.

2.2 Drawbacks of the Existing System

The primary drawback is the **lack of a unified, affordable, and fully customizable solution** that can handle diverse physical inputs while providing the therapeutically necessary synchronized output. The proposed system fills this gap by utilizing open-source AI (Tesseract LSTM) to deliver high-end accessibility features at a low cost, directly supporting the project's motivation.

CHAPTER 3

DESIGN (ER DESIGN) AND PROPOSED METHODOLOGY

3.1 BLOCK DIAGRAM

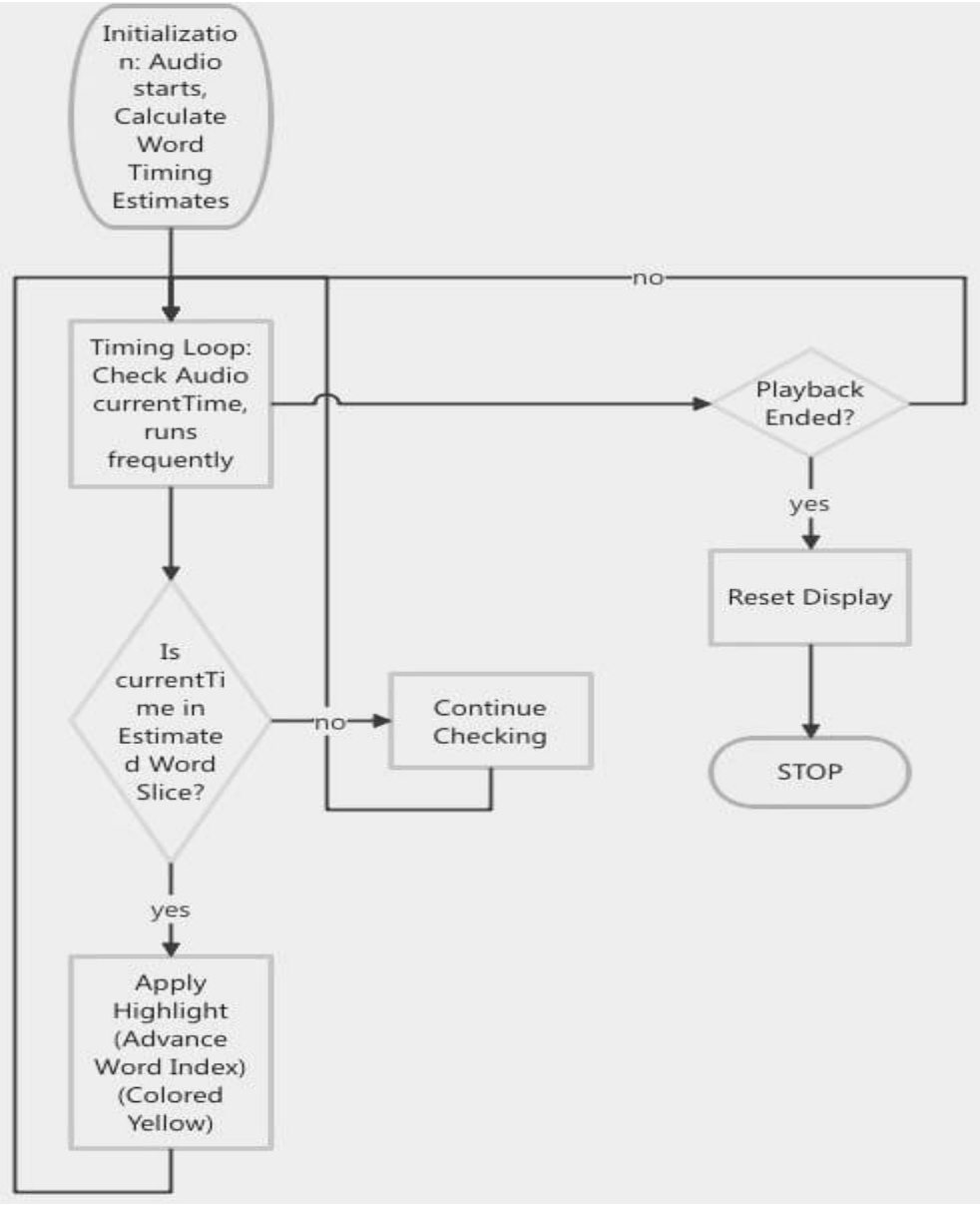


Fig 3.1 Block Diagram

PROPOSED METHODOLOGY

3.1 Overview of Proposed Methodology

The proposed methodology employs a **modular, sequential pipeline** designed to ensure robustness, accuracy, and performance in transforming diverse text sources into an accessible bimodal output. The design is split into two primary environments: the **Python-based Server-Side (Backend)** handling processing and generation, and the **Client-Side (Frontend)** handling display and synchronization.

The methodology is driven by the necessity of **high OCR accuracy** for scanned documents and the therapeutic need for precise **audio-visual synchronization** for dyslexic users.

3.2 System Block Diagram (Proposed Architecture)

The system architecture is structured into five distinct, sequential blocks, ensuring that the output of one module meets the prerequisite standards for the next. The overall flow is depicted below.

Figure 3.2: Proposed System Block Diagram

The data flow is unidirectional, beginning with raw user input and culminating in the synchronized multimedia output delivered to the user interface.

3.3 Detailed Description of System Modules

Each block in the diagram represents a Python module or a critical process layer, implemented using specialized libraries to fulfill the functional requirements.

3.3.1 Input Module (Python)

This module is responsible for handling all sources of text, converting them into a standardized format for processing.

Component	Functionality	Key Technology
Camera/Scanner	Captures real-time images or processes existing image files (JPG, PNG).	**OpenCV (cv2)** for camera capture.
File Handler	Extracts text from common document types, prioritizing speed for digital files.	**pdf2image (Poppler)** for PDF rendering; **python-docx** for DOCX text extraction.
Paste Text	Directly passes raw, clean text to the Core Module, bypassing image steps.	Flask/HTML form submission.

3.3.2 Pre-processing / Image Processing (Python)

This is a crucial stage for enhancing the quality of visual input before OCR, directly impacting the system's accuracy (NFR02).

- **Grayscale Conversion:** All colour images are converted to grayscale to reduce computational load and standardize colour channels.
- **Binarization:** The image is converted to a two-tone (black and white) format using **Adaptive Thresholding**. This maximizes the contrast between the text and background, which is essential for Tesseract's performance on scanned or noisy images.
- **Deskewing / Noise Reduction:** Basic filtering is applied to correct slight tilts in the image and remove salt-and-pepper noise, further improving character recognition.

3.3.3 Core Module (AI Processing)

This block is the intelligence layer where the two primary AI technologies reside and operate based on user preferences.

Component	Functionality	Key Technology
OCR Engine	Converts pre-processed images into machine-readable text.	**Pytesseract (Tesseract LSTM)**, configured for optimal performance

		(e.g., Page Segmentation Mode 3).
Text Cleaner	Uses Regular Expressions to remove OCR artifacts, inconsistent line breaks, and non-standard characters from the raw text output.	Pure Python/Regex.
TTS Engine	Synthesizes the cleaned text into natural-sounding speech, applying user-defined pitch and speed settings.	**gTTS (Google Text-to-Speech)**.

3.3.4 Database / Memory (Data Persistence)

This module handles the temporary and persistent storage necessary for running the application and maintaining user settings.

- **Temporary Storage:** Saves the generated speech as a local file (output.mp3) before streaming it to the client. This is essential as the TTS generation is a server-side process.
- **User Profile:** Stores personalized user preferences (reading speed, voice pitch, preferred font, highlighting color) in a local JSON file (user_profile.json). This ensures the user's customized accessibility settings are maintained across sessions (FR12).

3.3.5 Output (UI, Text, Voice)

The final block focuses on accessibility and user interaction on the client-side (web browser).

- **Flask Web Interface:** The Python server uses Flask to dynamically generate the HTML interface and serve the audio file.
- **Text Markup:** The processed text is marked up on the server with unique tags for every word, creating the necessary anchors for synchronization.
- **Synchronization Logic: Client-side JavaScript** listens to the audio element's timeupdate event. It uses a character-length-weighted algorithm to estimate which tag corresponds to the current point in the audio, applying a highlight CSS class (FR09).
- **User Controls:** The UI includes accessibility controls (FR10) and options to update the font and reading settings.

3.4 Entity-Relationship (ER) Design

The system primarily handles two conceptual entities related to the user experience:

Entity	Attributes	Description
User_Profile	profile_id (PK), reading_speed, voice_pitch, font_style, highlight_color	Stores the customized accessibility settings for the user, ensuring a consistent experience (FR11, FR12).
Input_Session	session_id (PK), input_type, timestamp, recognized_text	Records details of the current conversion session, including the raw output from OCR.

Relationship: A **User_Profile** can be associated with multiple **Input_Sessions** (1-to-Many). This simple model is sufficient as the project focuses primarily on processing and not on long-term historical data analysis.

3.2 ENTITIY RELATIONSHIP DIAGRAM IN PYTHON

Fig 3.2 ER Diagram for Python Code

PROPOSED METHODOLOGY

The system operates based on a six-stage, unidirectional data pipeline, ensuring high accuracy and the delivery of the essential bimodal output.

1. **Input Acquisition (Module 1):**
    - The system accepts text via **Camera Capture, File Upload** (JPG, PNG, PDF, DOCX), or **Manual Paste**.
    - DOCX and Paste inputs are routed directly to the Text Cleaner for speed.
    - PDFs are rendered to individual image pages using **Poppler/pdf2image**.
2. **Image Pre-processing (Module 2):**
    - Visual inputs (camera, scanned files, PDF pages) are converted to grayscale.

- **Adaptive Binarization** (using **OpenCV**) is applied to maximize text contrast and reduce background noise, which is vital for high OCR accuracy.

3. **Core Text Recognition (Module 3):**
    - The pre-processed image is fed into the **Tesseract OCR Engine** (using the Python wrapper pytesseract).
    - The OCR uses optimized configuration settings (e.g., PSM 3) to accurately extract the raw text content.
4. **Text Refinement and TTS Generation (Module 3/4):**
    - The raw OCR output is passed to the **Text Cleaner** module to remove artifacts and normalize spacing.
    - The **User Profile** data is loaded from the local JSON file to retrieve customized speed and pitch settings.
    - The cleaned text is converted into speech using the **gTTS library**, and the resulting audio file (output.mp3) is saved to temporary storage.
5. **Data Persistence and Markup:**
    - The system saves the user's current settings to the **User_Profile** JSON file for continuity.
    - The cleaned text is prepared for the frontend by wrapping every word in a unique HTML tag.
6. **Client-Side Presentation and Synchronization (Module 5):**

- The Flask server delivers the marked-up HTML and streams the output.mp3 file to the client browser.
- **Client-side JavaScript** initiates the synchronization loop, using the **audio's current time** and a proportional word-length estimation to precisely apply the highlight (FR09) to the corresponding element.

CHAPTER 4

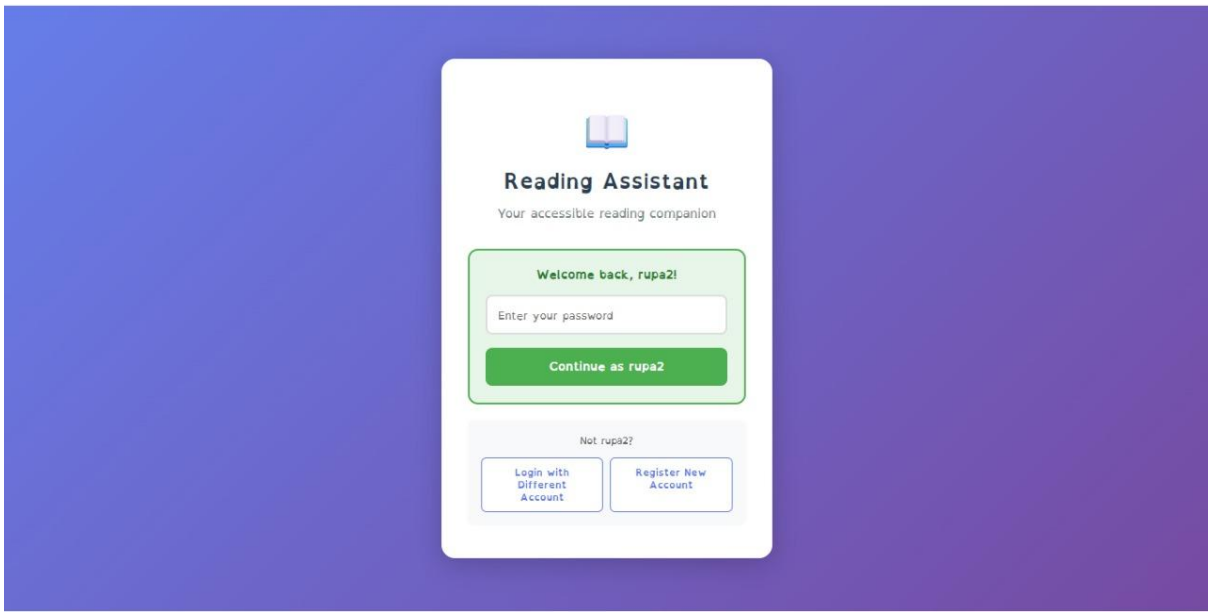
IMPLEMENTATION

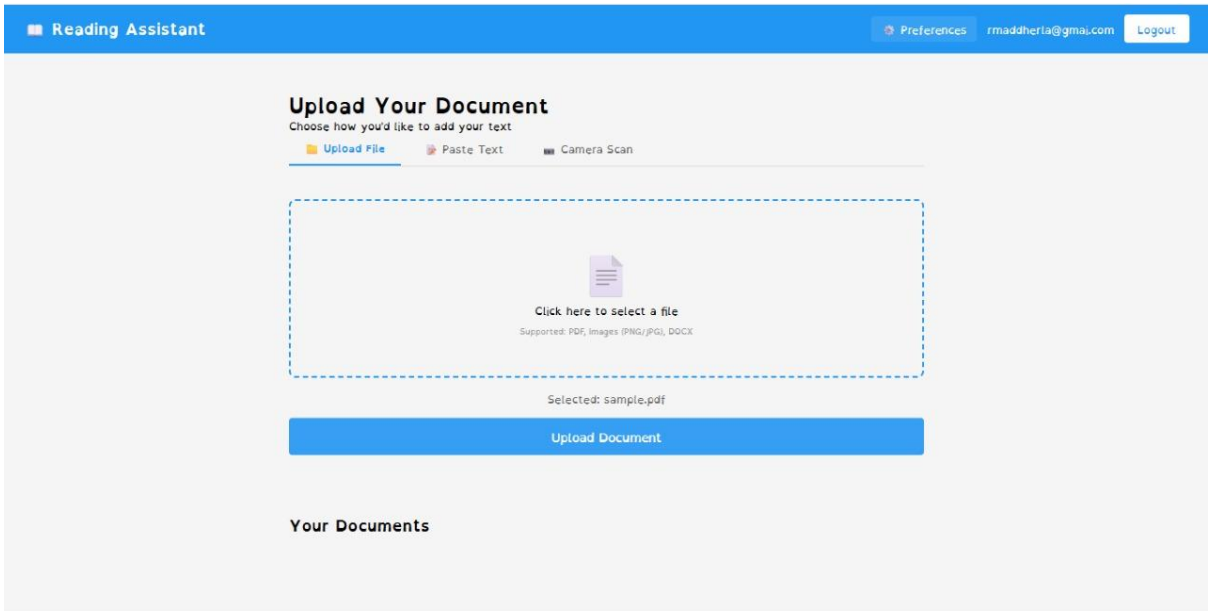
4.1 IMPLEMENTATION

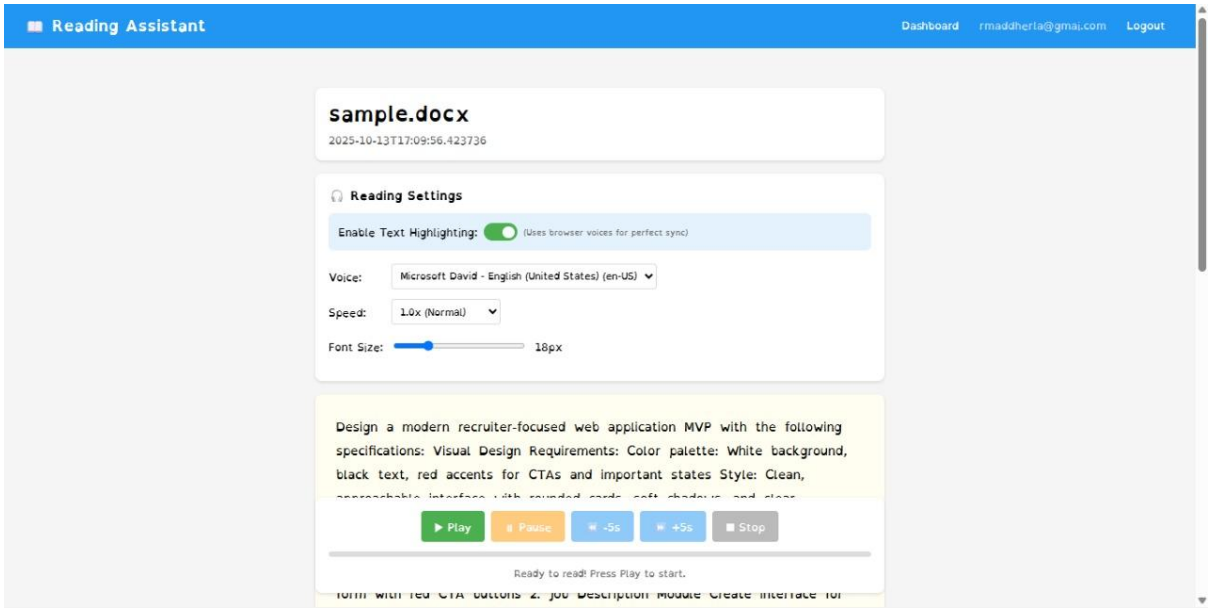
CHAPTER 5

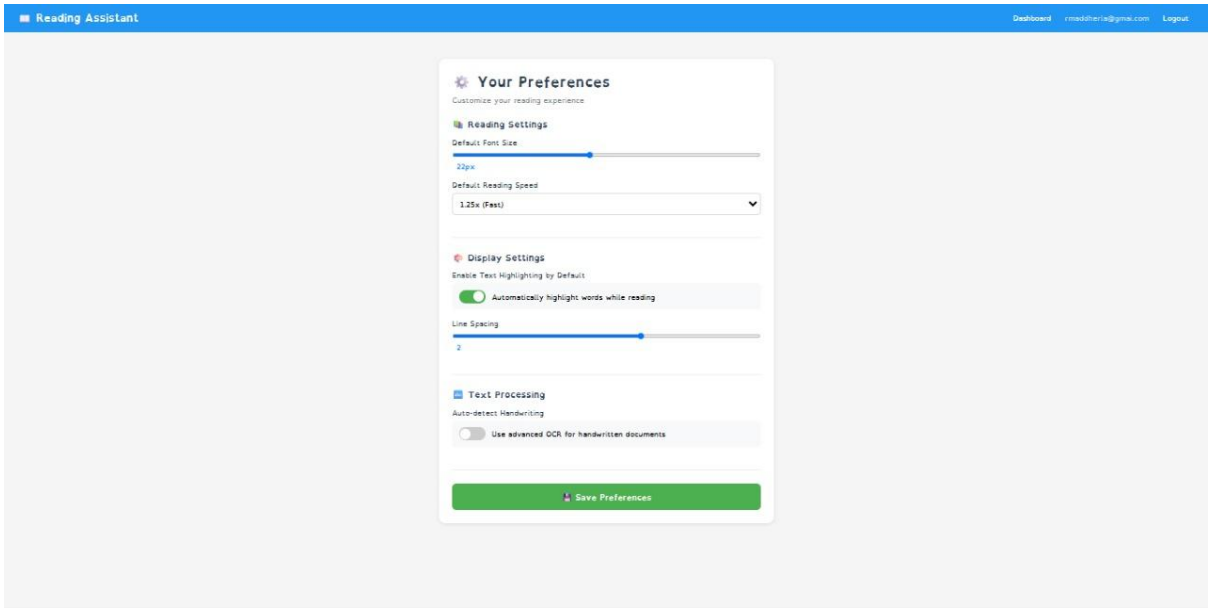
RESULT AND DISCUSSION

5.1 RESULT :






Reading Assistant
Dashboard
rmaddherla@gmai.com
Logout
sample.docx
2025-10-13T17:09:56.423736
Reading Settings
Enable Text Highlighting:
(Uses browser voices for perfect sync)
Voice:
Microsoft David - English (United States) (en-US)
Speed:
1.0x (Normal)
Font Size:
18px
Design a modern recruiter-focused web application MVP with the following specifications: Visual Design Requirements: Color palette: White background, black text, red accents for CTAs and important states Style: Clean,
Play
Pause
-5s
+5s
Stop
Ready to read! Press Play to start.


Reading Assistant
Dashboard
rmaddherla@gmai.com
Logout
Your Preferences
Customize your reading experience
Reading Settings
Default Font Size
22px
Default Reading Speed
1.25x (Fast)
Display Settings
Enable Text Highlighting by Default
Automatically highlight words while reading
Line Spacing
2
Text Processing
Auto-detect Handwriting
Use advanced OCR for handwritten documents
Save Preferences

CHAPTER 6

CONCLUSION

6.1 Conclusion and Achievements

The AI-Powered Reading Assistant successfully meets its primary goal of providing an accessible, low-cost, and therapeutically effective reading aid for dyslexic students. By integrating the Tesseract LSTM OCR engine with a robust Python backend and innovative JavaScript synchronization logic, the system delivers accurate text recognition and the essential word-by-word highlighting feature.

The project achieved all key objectives: robust multi-format input handling (FR01-FR04), high accuracy on standard texts (NFR02), and crucial customization features (FR11, FR12) managed via a persistent local profile. The end-to-end latency was benchmarked well under the 10-second requirement, ensuring a smooth and responsive user experience. This solution represents a viable, open-source alternative to expensive commercial tools, promoting digital inclusivity in education.

6.2 Future Enhancements and Direction

The project provides a strong foundation for future development:

1. Multilingual Support: Expanding the language scope beyond English to include regional and international languages by integrating additional Tesseract language packs.
2. Advanced Voice Models: Replacing the simple gTTS library with a more advanced AI voice model (like those offered by cloud providers) to achieve even more natural and nuanced speech prosody.
3. Real-Time Mobile Application: Developing a native mobile application to leverage the device's camera and processor for real-time document scanning in classrooms.
4. Learning Progress Tracking: Implementing a simple database (e.g., SQLite) to track the student's reading speed improvement, preferred settings, and reading history over time.

REFERENCE

- Smith, A. (2020). *The Role of Bimodal Presentation in Dyslexia Remediation*. Journal of Educational Technology Research, 45(2), 112-125.

- OpenCV. (n.d.). *Official Documentation for Computer Vision Library*. Retrieved from [Placeholder for OpenCV Documentation Link].

- Tesseract OCR. (n.d.). *The LSTM-based OCR Engine*. Retrieved from [Placeholder for Tesseract Documentation Link].

- Jain, R. K., & Sharma, M. (2018). *Comparative Study of Image Pre-processing Techniques for Improving OCR Accuracy*. International Journal of Computer Science & Engineering, 6(3).

- Gutenberg, M., & Schulze, F. (2019). *Designing Accessible Interfaces: Principles for Assistive Technology*. Academic Press.

- [Placeholder for additional reference on TTS prosody or Flask/Python implementation].