Accessible Reading Assistant - Product Specification Document

1. Overview

The **Accessible Reading Assistant (ARA)** is a web-based application designed to make complex documents easier to understand for diverse users, including students, professionals, non-native speakers, and individuals with reading disabilities. Users can upload documents, which will be converted into simplified text and optionally read aloud. The platform emphasizes **privacy and confidentiality**: no user data or uploaded documents are stored.

2. Goals and Objectives

- **Simplification**: Convert uploaded text into an easy-to-read format while preserving meaning.
- **Accessibility**: Provide a read-aloud feature to assist users with visual impairments or reading challenges.
- **Privacy-first**: Ensure zero data persistence. No user accounts, no document history.
- **Usability**: Simple, clean UI that works on web and mobile.
- **Extensibility**: Future integrations with AI models (e.g., Azure AI Foundry, OpenAI) without compromising privacy.

3. Key Features

1. **Document Upload**
2. Supported formats: PDF, DOCX, TXT.
3. Single-file upload per session.
4. Size limit: TBD (recommend 10MB).
5. **Text Simplification**
6. Convert complex sentences into simpler language.
7. Reduce jargon and improve readability.
8. Highlight simplified vs. original text (optional toggle).
9. Use AI-based summarization/paraphrasing models.
10. **Text-to-Speech (TTS)**
11. Read simplified text aloud.
12. Options for playback controls: Play, Pause, Stop.
13. Adjustable speech rate and voice (male/female).
14. **Copy & Export**
15. Allow users to copy simplified text.
16. Download as plain text (.txt).

17. No server-side storage.

18. **Confidentiality by Design**

19. No login/account system.
20. No storage of uploaded documents or converted text.
21. All processing happens in memory during a session.

4. Non-Goals

- No long-term storage or user profiles.
- No collaborative features (e.g., shared notes, multi-user editing).
- No advanced document editing beyond simplification/export.

5. User Flow

1. **Landing Page**: Brief intro, upload prompt.
2. **Upload Document**: Drag-and-drop or select file.
3. **Processing**: AI model simplifies text, generates output.
4. **Results Page**:
5. Display simplified text.
6. Read aloud option.
7. Copy/download buttons.
8. **End of Session**: Data cleared once tab is closed.

6. Architecture

- **Frontend**: React.js + Tailwind (lightweight, responsive).
- **Backend**: Node.js/Express API.
- **AI Integration**: Azure AI Foundry / OpenAI API for text simplification and TTS.
- **Security**:
- HTTPS only.
- Ephemeral in-memory processing.
- No logs of user content.

7. Privacy & Compliance

- No user data stored.
- No PII collection.
- Compliance with GDPR principles (right to erasure is inherent since no storage).
- Strong encryption in transit (TLS 1.2+).

8. Success Metrics

- **Usability**: Users can process documents end-to-end in <2 minutes.
- **Accuracy**: Simplified text retains >90% of meaning.
- **Adoption**: Target of 500 unique users in pilot.
- **Accessibility**: WCAG 2.1 compliance for UI and TTS.

9. Risks and Mitigation

- **Risk**: Misinterpretation of simplified text.
- *Mitigation*: Provide side-by-side original vs. simplified.
- **Risk**: Large file uploads causing latency.
- *Mitigation*: Set file size limits; chunk processing.
- **Risk**: AI dependency downtime.
- *Mitigation*: Fallback to rule-based simplification.

10. Roadmap (Phased)

Phase 1: MVP - File upload (PDF, TXT, DOCX). - Simplification engine (AI-based). - TTS playback. - Copy/export text.

Phase 2: Enhancements - Side-by-side comparison (original vs simplified). - Voice customization. - Mobile optimization.

Phase 3: Future - Multilingual support. - On-device processing for extra security. - Browser extension version.

11. Open Questions

- What readability level should simplification target? (e.g., 8th grade reading level).
- Should there be support for non-English languages from the start?
- Will TTS use server-based models or client-side APIs (e.g., Web Speech API)?