EdTech Accessibility Hub for Dyslexic Students

Team CodeForBharat

June 2025

Abstract

The EdTech Accessibility Hub for Dyslexic Students is a website designed to convert textbooks into dyslexia-friendly formats, including specialized fonts and audio, with AI-powered summarization. This project aims to bridge the gap in EdTech inclusivity and support dyslexic learners in India.

Concept

The concept involves creating a website that converts standard textbooks into formats that are accessible for dyslexic students. This includes converting text to dyslexia-friendly fonts and providing audio versions of the text, along with AI-generated summaries.

Competitive Edge

- Targets the intersection of EdTech and inclusivity.
- Utilizes Generative AI for content adaptation.
- Scalable with cloud storage solutions.
- Potential impact on 70M+ dyslexic learners in India.

Constraints

- Limited hardware: 1 AMD Ryzen with RTX 3050 or free Colab.
- Must build a fully scalable website (no offline/SMS/MMS/WhatsApp).

Core Features

1. **Text Conversion to Dyslexia-Friendly Formats**

- Font Conversion: Convert standard text to OpenDyslexic font.
- Audio Conversion: Text-to-speech (TTS) for auditory learning.
- AI Summarization: Provide concise summaries of textbook content.

2. **User Management**
    - Upload textbooks (PDF, DOCX, images).
    - Save converted materials for later access.
3. **Scalable Web Platform**
    - Frontend: Responsive and accessible.
    - Backend: Handle file uploads, processing, and storage.

Technical Planning

Frontend

- **Framework**: Next.js (React) for server-side rendering and static site generation.
- **UI Library**: Tailwind CSS for rapid development.
- **Accessibility**: Ensure the website itself is accessible (ARIA tags, keyboard navigation, high contrast mode).

Backend

- **Framework**: Flask (Python) or FastAPI.
- **Cloud Storage**: Firebase Storage or AWS S3 (free tier) for storing user files.
- **Database**: Firebase Firestore (NoSQL) for user data and metadata about conversions.

AI Components

1. **Text Extraction from Uploaded Files**
    - Use PyPDF2 for PDFs and python-docx for DOCX.
    - For images, use Tesseract OCR (open-source).
2. **Dyslexia-Friendly Font Conversion**
    - Replace the font in the extracted text with OpenDyslexic and re-render.

3. **Text-to-Speech (TTS)**
   - Use pyttsx3 (offline, cross-platform) or Coqui TTS (open-source).
4. **AI Summarization**
   - Use a distilled version of BERT (e.g., DistilBART) for extractive summarization.

Scalability and Performance

- **Background Processing**: Use Celery with Redis for task queues.
- **Caching**: Cache summaries and TTS outputs for the same content to avoid reprocessing.
- **Hosting**: Frontend on Vercel, backend on a cloud service (like Render or Heroku) or use serverless functions.

Implementation Roadmap

Week 1: Setup and Basic Functionality

- Set up Next.js frontend with a simple upload form.
- Set up Flask backend with file upload endpoint.
- Implement text extraction from PDFs and DOCX.
- Implement font conversion.

Week 2: AI Integration

- Integrate DistilBERT for summarization using Hugging Face `transformers`.
- Integrate TTS: Start with pyttsx3 for MVP, then move to Coqui TTS if time allows.
- Implement task queue (Celery) for background processing.

Week 3: User Management and Polishing

- Add Firebase for user authentication and storage.
- Implement saving converted materials and history.
- Optimize for scalability: caching, load testing.

Critical Links

Resource	Link
PyPDF2	https://pypi.org/project/PyPDF2/
python-docx	https://pypi.org/project/python-docx/
Tesseract OCR	https://github.com/tesseract-ocr/tesseract
OpenDyslexic Font	https://opendyslexic.org/
pyttsx3	https://pypi.org/project/pyttsx3/
Coqui TTS	https://github.com/coqui-ai/TTS
DistilBART Model	https://huggingface.co/sshleifer/distilbart-cnn-12-6
Next.js	https://nextjs.org/
Tailwind CSS	https://tailwindcss.com/
Flask	https://flask.palletsprojects.com/
Firebase	https://firebase.google.com/
Celery	https://docs.celeryq.dev/
Redis	https://redis.io/
Vercel	https://vercel.com/

Hardware Optimization Tips

- **VRAM Management**: Set `max_split_size_mb=512` in PyTorch.
- **Parallel Processing**: Use Celery with 4 workers (1 per CPU core).
- **Freeze Layers**: For DistilBART fine-tuning:

```
for param in summarizer.model.parameters():
    param.requires_grad = False  # Last 2 layers
        only
```

Getting Started

Prerequisites

- Python 3.10
- Node.js
- Docker
- Conda (for environment management)

Setup

1. **Create Conda Environment**

```
conda create -n dyslexia python=3.10
conda activate dyslexia
```

2. **Install Python Dependencies**

```
    pip install torch torchvision torchaudio --index-
       url https://download.pytorch.org/whl/cu118
    pip install transformers bitsandbytes accelerate
    pip install pytesseract pdf2image python-docx
       PyPDF2
    pip install flask celery redis
```

3. **Install Tesseract OCR**

```
    sudo apt install tesseract-ocr  # Linux
```

4. **Set Up Next.js Frontend**

```
    npx create-next-app@latest
```

5. **Set Up Firebase**

   - Create a Firebase project at `https://firebase.google.com/`.
   - Set up Firebase Storage and Firestore.

6. **Run Redis and Celery**

```
    docker run -d -p 6379:6379 redis/redis-stack-
       server:latest
    celery -A tasks worker --loglevel=info --pool=
       solo
```

Conclusion

This project aims to create a scalable and efficient EdTech Accessibility Hub for dyslexic students, leveraging modern web technologies and AI to provide an inclusive learning experience. By following the outlined steps and utilizing the provided resources, you can develop a fully functional and impactful solution within a week.