
Project: Multifunctional File Conversion and AI-Powered Services

1. Overview

The project aims to create a comprehensive platform with two core modules:

1. **File Conversion Module**: Converts files between different formats (e.g., PDF to DOCX, MP3 to WAV).
2. **AI-Powered Module**: Leverages machine learning for tasks like speech-to-text and text-to-audio conversions.

The platform will be designed with a user-friendly **UI/UX** using **Figma** and supported by robust back-end technologies.

2. Technologies and Their Roles

Frontend

- **HTML/CSS/JavaScript**: For structuring, styling, and adding interactivity to the web pages.
- **Figma**: To prototype and design the UI/UX layout.

Backend

- **Node.js**: Core framework for running JavaScript on the server.
- **Express.js**: Simplifies API creation for file conversion and AI tasks.
- **Python**: For machine learning models in the AI-powered module.
- **Libraries**:
    - **Sharp**: Image processing.
    - **FFmpeg**: Audio and video format conversions.
    - **Speech-to-Text API**: Transcribe audio to text.
    - **Text-to-Speech**: Generate audio from text.

Database

- **MongoDB**: Store user data, logs, and other information.

- **3. Directory Structure**
- plaintext
- Copy code
- project-root/
- │
- ├── file-conversion-module/
- │   ├── index.js   # Main file for file conversion
- │   ├── converter.js   # Functions for file format conversion
- │   ├── utils.js   # Utility functions for processing
- │
- ├── ai-conversion-module/
- │   ├── index.py   # Main AI logic
- │   ├── speech_to_text.py # Speech-to-text implementation
- │   ├── text_to_audio.py # Text-to-speech functionality
- │
- ├── server/
- │   ├── server.js   # Backend server setup
- │   ├── routes/
- │   │   ├── fileRoutes.js # Routes for file conversion APIs
- │   │   ├── aiRoutes.js   # Routes for AI APIs
- │   ├── middleware/
- │   │   ├── auth.js   # Authentication middleware
- │   ├── config.js   # Environment configurations
- │
- ├── public/
- │   ├── index.html   # UI entry point
- │   ├── styles.css   # Styles for the UI
- │   ├── app.js   # Frontend logic
- │
- ├── database/
- │   ├── db.js   # Database connection setup
- │   ├── models/
- │   │   ├── User.js   # User schema
- │   │   ├── FileLog.js   # File processing logs
- │
- ├── tests/
- │   ├── testFileConversion.js # Test cases for file conversion module
- │   ├── testAIConversion.py   # Test cases for AI module
- │
- ├── package.json   # Node.js dependencies
- ├── requirements.txt   # Python dependencies
- ├── README.md   # Documentation

4. Setting Up Development Environment

Follow these steps:

Frontend

1. Install a code editor like **VS Code**.
2. Use **Figma** to create the design for your application.

Backend

1. Install **Node.js** and **Python**.
2. Set up a package manager:
    - Use **npm** for Node.js dependencies.
    - Use **pip** for Python libraries.

Database

1. Install and configure **MongoDB**.

Project Directory

1. Create directories using VS Code's terminal with these commands:

    ```
    mkdir project-root

    cd project-root

    mkdir file-conversion-module ai-conversion-module server public database tests

    mkdir server/routes server/middleware database/models
    ```

2. Create files in each folder (use echo. for Windows or touch for Mac/Linux).

5.Tools Required

- **VS Code**: Code editor for full-stack development.
- **Node.js and npm**: Server-side and library management.
- **Python and pip**: Machine learning and AI modules.
- **MongoDB**: Database for user and file data.
- **Figma**: UI/UX design
- **Postman**: Test your APIs.

6. Workflow

1. **UI/UX Design**:
    - Use Figma to design the interface, ensuring responsiveness and user-centric layouts.
2. **Back-End Development**:
    - Develop REST APIs for both modules.
    - Use Sharp for image processing and FFmpeg for video/audio conversions.
    - Integrate AI functionalities with Python and APIs like Google's Speech-to-Text.
3. **Frontend Development**:
    - Build the UI using HTML, CSS, and JavaScript.
    - Integrate APIs to enable interaction between the frontend and backend.
4. **Testing**:
    - Write unit tests for Node.js and Python.
    - Use tools like Postman for API testing.

7. Deployment Plan

- Use **Docker** to containerize the application.
- Deploy on platforms like **AWS**, **Heroku**, or **Azure** for scalability.