RapidFort: Campus Recruitment Drive

Word-to-PDF Conversion Web Application

Comprehensive Documentation

Developer Name: **Dev Inder Garg**

Project Start Date: **21-11-2024**

Project Completion Date: **23-11-2024**

Repository Link: https://github.com/Devgarg1302/Word-pdf/

Frontend Hosted Endpoint: https://word-pdf.vercel.app/

Backend Hosted Endpoint: https://word-pdf-v08j.onrender.com

Table of Contents

1 INTRODUCTION

1.1 Overview

The Word-to-PDF Conversion Web Application is a lightweight and user-friendly solution for processing Word documents (.docx). It allows users to upload .docx files, view their metadata, convert them into PDF format, and securely download the converted files. Designed with simplicity and performance in mind, the application ensures a seamless user experience.

1.2 Objective

The primary goal of this project is to provide an efficient, secure, and scalable web application for converting Word documents to PDF. This application can serve individuals and businesses looking to handle document conversion with features like password protection, microservice architecture, and containerized deployment.

1.3 Scope

This project focuses on the following:

- Uploading .docx files.
- Displaying basic metadata, such as file size and creation date.
- Converting .docx files to PDF with optional password protection.
- Providing a simple and intuitive user interface for seamless interaction.

- Implementing modern software practices, such as microservices, containerization (Docker), and Kubernetes deployment.

While the application supports .docx file conversion, future extensions may include other file formats and advanced features such as bulk conversion and cloud storage integration.

This documentation outlines the features, architecture, and deployment of the application, providing developers and users with a comprehensive guide to understanding and utilizing the system effectively.

2 FEATURES

The Word-to-PDF Conversion Web Application provides a robust set of features to deliver a seamless and efficient user experience, backed by modern software practices for scalability and reliability.

1. File Upload

- Allows users to upload Word documents (.docx) through an intuitive and responsive interface.
- Validates file format and size to ensure compatibility with the conversion process.
- Provides real-time feedback on upload status, with error messages for invalid inputs.

2. Metadata Display

- Extracts and displays key metadata from the uploaded .docx file, including:
    - File name
    - File size
    - Last modified date
    - Author Name
- Enhances user understanding of the document before conversion.

3. PDF Conversion

- Converts uploaded .docx files into high-quality PDFs using the libreoffice-convert library.
- Supports diverse document content, including text, images, and tables, ensuring accuracy and efficiency.

4. Password Protection

- Offers the ability to secure the generated PDF with a custom password for confidentiality.
- Ensures enhanced document security during downloads and future sharing.

5. Download Functionality

- Allows users to download the converted PDF directly from the application.
- Ensures efficient and reliable downloads with minimal latency.

6. Hosting and Scalability

- **Frontend** is hosted on **Vercel**, providing global availability and rapid response times.
- **Backend** is deployed on **Render**, ensuring high availability and seamless integration with the frontend.
- Both deployments support continuous updates through automated pipelines.

7. Microservice Architecture

- Designed with a modular architecture separating frontend and backend services.
- Each service is containerized using Docker, ensuring consistency and flexibility across environments.

8. CI/CD Pipeline

- Uses **GitHub Actions** to automate the build, test, and deployment process.
- Automatically builds Docker images and pushes them to **Docker Hub**, enabling faster and more reliable updates.

9. User-Friendly Interface

- A modern and clean UI built with React.js ensures easy navigation.
- Simplifies interactions, including file uploads, metadata viewing, and managing conversion settings.

This feature set makes the application a comprehensive and scalable solution for Word-to-PDF conversion, catering to both individual and enterprise needs.

3 TECHNOLOGY STACK

Frontend

- **React.js**
  - Used to build an interactive and user-friendly interface for file uploads and interactions.
  - Ensures a seamless experience for viewing metadata, initiating conversions, and downloading the output.

- **Hosting Platform:**
  - **Vercel**: A modern platform for hosting the frontend, offering automatic deployments, scalability, and a global CDN for fast delivery.

Backend

- **Node.js**
  - The core runtime for handling server-side operations, including file uploads and API endpoints.

- **Express.js**
  - A lightweight framework used to manage routes, handle requests, and facilitate communication between the frontend and backend.

- **Document Conversion Library:**
  - **libreoffice-convert**: Powers the high-quality conversion of Word documents (.docx) to PDF, ensuring efficient and accurate processing.

- **Hosting Platform:**

- **Render**: A developer-friendly platform chosen for backend hosting, offering simplicity, reliability, and support for continuous deployment.

Containerization

- **Docker**
  - Both the frontend and backend are containerized, ensuring consistent behaviour across environments and simplifying deployment workflows.

CI/CD Integration

- **GitHub Actions**
  - Automates the CI/CD pipeline, including:
    - Building Docker images.
    - Pushing Docker images to Docker Hub.
    - Deploying updates to hosting platforms.

Development and Testing Tools

- **Version Control:** Git (hosted on GitHub).
- **Postman:** Used for testing API endpoints during development.
- **ESLint and Prettier:** Maintains code quality and enforces consistent formatting.

4 SYSTEM ARCHITECTURE

The system architecture for the Word-to-PDF Conversion Web Application is designed to ensure modularity, scalability, and maintainability. The architecture adopts a microservices approach, where each component is loosely coupled, facilitating ease of development and deployment.

4.1 Overview

The architecture consists of two primary components:

- **Frontend:** A React-based client application for user interaction.
- **Backend:** A Node.js-based server for handling document processing and conversion.

These components communicate via RESTful APIs, with backend services containerized and orchestrated using Docker and Kubernetes.

4.2 Architecture Components

Frontend

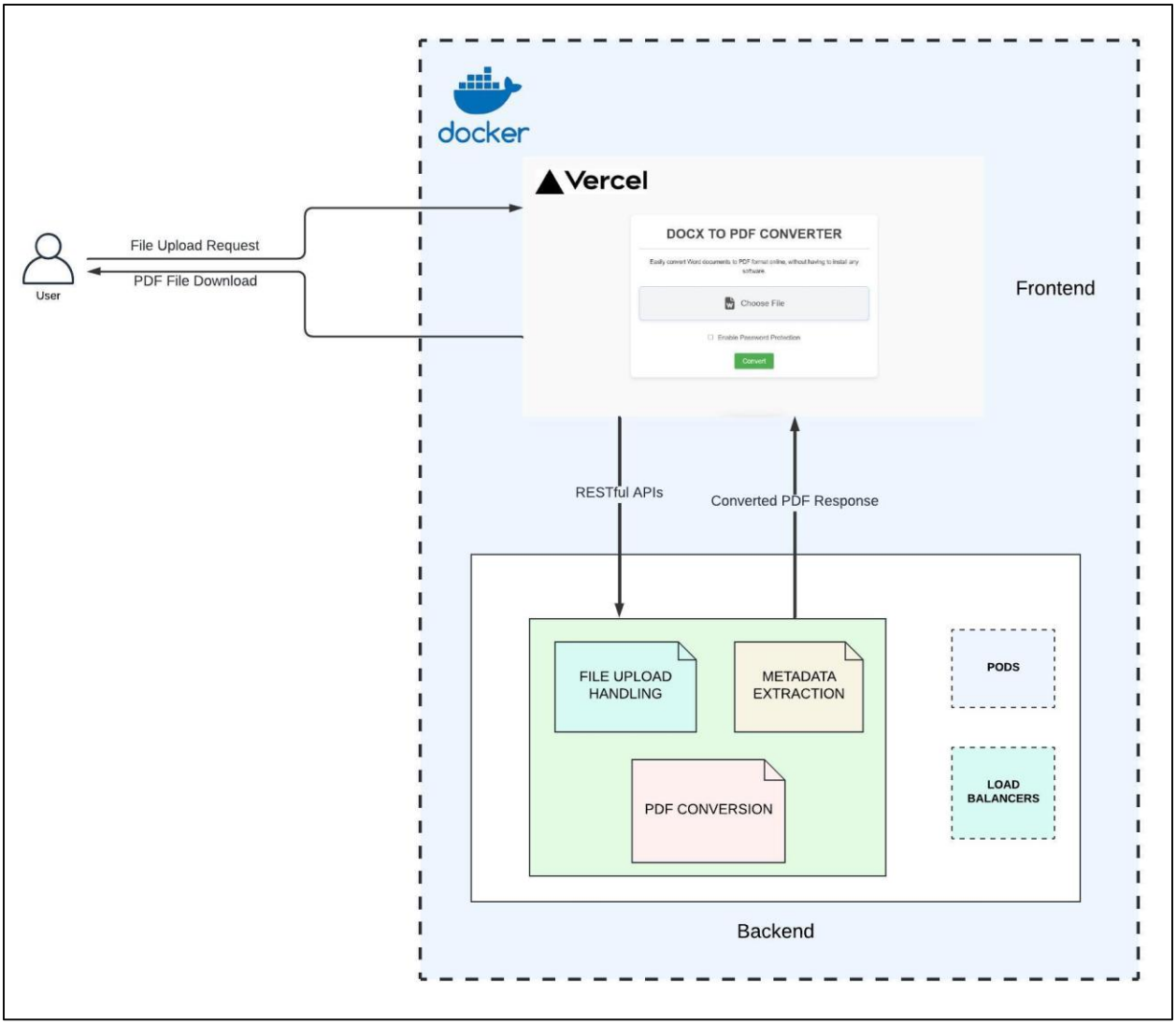
- **Role:** Provides an interactive UI for uploading Word documents, viewing metadata, and downloading converted PDFs.
- **Technologies Used:**
  - React.js for building the interface.
  - Axios for API communication.
- **Deployment:** Hosted on Vercel for low-latency, globally distributed content delivery.

Backend

- **Role:** Handles file uploads, extracts metadata, and processes document conversion to PDF.
- **Key Functionalities:**
  - File validation and metadata extraction.
  - PDF generation using libreoffice-convert.
  - Error handling and response management.
- **Technologies Used:**
  - Node.js and Express.js for API development.
  - Docker for containerization.
- **Deployment:** Hosted on Render with auto-scaling and managed services.

Document Conversion

- **Library:** libreoffice-convert ensures high-quality Word-to-PDF conversion while maintaining the original formatting and content integrity.


docker
Vercel
DOCX TO PDF CONVERTER
Choose File
Convert
Frontend
User
File Upload Request
PDF File Download
RESTful APIs
Converted PDF Response
FILE UPLOAD HANDLING
METADATA EXTRACTION
PDF CONVERSION
PODS
LOAD BALANCERS
Backend

5 API ENDPOINTS

The API endpoints for the Word-to-PDF Conversion Web Application are structured to handle file uploads, conversions, and error handling effectively. Below is the detailed documentation of the endpoints and how they interact with the frontend.

1. Base URL

The backend API is hosted on Render, with the base URL: https://word-pdf-v08j.onrender.com

2. API Endpoints

1. Upload and Convert Document

- **Endpoint:** /convertFile
- **Method:** POST
- **Description:** Uploads a .docx file and converts it to a PDF. Returns the converted PDF as a downloadable file.
- **Request Details:**
    - **Headers:**
        - "Content-Type": "multipart/form-data"
    - **Body:**

        - formData (multipart/form-data): Contains the .docx file.

- **Response:**
    - **Success (200):** Returns the converted PDF file as a binary blob.
        - Example Frontend Code to Handle Response:
        - const url = window.URL.createObjectURL(new Blob([response.data]));
        - const link = document.createElement("a");
        - link.href = url;
        - link.setAttribute("download", `${file.name.split(".")[0]}.pdf`);
        - document.body.appendChild(link);
        - link.click();
        - link.parentNode.removeChild(link);
    - **Error Responses:**
        - 400 Bad Request: Invalid file or missing input.
        - 500 Internal Server Error: If the conversion fails.
    - Example Error Handling in Frontend:
    - catch (err) {
    - console.error("Error uploading file:", err.response?.data || err.message);

- alert("Failed to upload and convert the file.");
- }

2. Metadata Extraction

- **Description:** Retrieves metadata for the uploaded document.
- **Request Parameters:**
  - fileId (optional, string): The ID of the uploaded file for retrieving specific metadata.
- **Response:**
  - **Success (200):** Returns metadata like file name, size, upload date, and last modified date.
    - Example Response:
    - {
    - "fileName": "example.docx",
    - "fileSize": "2.5 MB",
    - "authorName": "xyz",
    - "lastModified": "2024-11-20"
    - }
  - **Error Responses:**

    - 404 Not Found: If metadata for the file is unavailable.

3. Error Handling

- All endpoints are designed to provide clear and consistent error messages.
- **Common Error Codes:**
    - **400 Bad Request:** The file is missing or invalid.
    - **404 Not Found:** Resource or file not found.
    - **500 Internal Server Error:** Internal issues during processing.

4. Key Highlights

- **Seamless Integration:** The /convertFile endpoint directly handles both file upload and conversion, simplifying the client-server interaction.
- **Binary Data Handling:** The PDF file is returned as a blob to the frontend for immediate download.
- **Error Awareness:** Client-side error handling ensures users are informed of any issues during upload or conversion.

This API endpoint structure provides a streamlined approach for processing documents while being adaptable for future enhancements, such as supporting multiple file formats or adding security features like password-protected PDFs.

6 USER GUIDE

Welcome to the **Word-to-PDF Conversion Web Application**! This user guide will walk you through how to interact with the application, upload your documents, convert them to PDF, and download the converted files.

1. Getting Started

To begin using the application, follow these steps:

1. **Open the Web Application:**
    - Visit the application URL in your browser: https://word-pdf.vercel.app/
2. **User Interface Overview:**

    The interface consists of:

    - **File Upload Section:** To upload your Word document.
    - **Progress Section:** To view the status of your file processing (will be displayed after file upload).

2. Uploading a Document

1. **Locate the File Upload Section:**

- You will see a file input area labeled **"Upload a Word Document (.docx)"**.

2. **Select Your Word Document:**
   - Click the **"Choose File"** button.
   - Browse through your local files and select the .docx file you wish to upload.
3. **Upload the File:**
   - After selecting the file, click the **"Upload and Convert"** button.
   - The application will send the file to the server for processing. During this process, you will see a **"Processing..."** message or progress indicator.

3. File Conversion

1. **Conversion Process:**
   - Once the file is uploaded, the backend will automatically start converting the .docx file into a PDF.
   - The application will process the file, and you will receive a notification once the conversion is completed.
2. **Download the Converted PDF:**

- Once the conversion is complete, the file will automatically be downloaded.

3. **Password Protection (Optional):**
    - If you want to add password protection to your PDF, you can enter a password in the provided field before clicking **"Upload and Convert"**.
    - The converted PDF will be secured with the password you provide, and you will be required to enter the password to view the file.

4. Viewing Metadata (Future Feature)

1. **Retrieve Document Metadata:**
    - After uploading a file, you will be able to view metadata about the document such as file name, size, and last modified date.

5. Troubleshooting

If you encounter any issues, here are some common solutions:

- **Invalid File Type:**
    - The application only supports .docx files. Make sure the document you upload is in the correct format.
    - **Solution:** Ensure your file is a .docx format before uploading.

- **Conversion Errors:**
    - If the conversion process fails or takes too long, try uploading a smaller file or check your internet connection.
    - **Solution:** Refresh the page, check for any error messages, and try uploading the file again.
- **No Download Button Appears:**
    - If the download button doesn’t show up after the file is uploaded, it could be due to a server error or network issue.
    - **Solution:** Check your browser’s developer tools for error messages or try again after a short wait.

7 FUTURE ENHANCEMENTS

The current implementation of the Word-to-PDF Conversion Web Application provides a solid foundation for document processing and conversion. However, several potential enhancements can further improve the application's usability, functionality, and scalability. Below are some planned features and improvements:

1. Support for Additional File Formats

- **Description:** Extend the application to support other file formats, such as:
    - Word files: .doc, .rtf
    - Presentation files: .pptx, .ppt
    - Spreadsheets: .xlsx, .xls
    - Text and image formats: .txt, .png, .jpeg
- **Benefit:** Allows a wider range of document conversions, increasing the application's versatility and user base.

2. Enhanced PDF Features

- Add options to:
    - Set custom page size and orientation (A4, Letter, Landscape, etc.).
    - Merge multiple .docx files into a single PDF.

    - Add watermarks (text or image-based) to PDFs during conversion.
    - Compress PDF files to reduce file size while maintaining quality.
- **Benefit:** Provides users with more control over their PDF output, enhancing customization.

3. Metadata Viewer

- **Description:** Implement a feature to view and edit metadata for uploaded documents and converted PDFs. Metadata may include:
    - Document title, author, subject, and keywords.
    - Creation and modification dates.
- **Benefit:** Enables users to manage their document properties effectively, particularly useful for professionals and businesses.

4. User Authentication and File Management

- **Description:** Introduce a user authentication system to allow registered users to:
    - Manage uploaded and converted files through a personal dashboard.
    - View the history of processed files.
- **Benefit:** Adds a layer of personalization and security, making the application more professional and reliable.

5. Advanced Security Features

- **Description:** Enhance the security of converted PDFs by adding features such as:
    - AES-256 encryption for password-protected PDFs.
    - Access restrictions: Prevent copying, printing, or editing of PDFs.
    - Digital signatures to verify the authenticity of PDFs.
- **Benefit:** Increases the trustworthiness and security of documents for business and personal use.

6. Scalability Enhancements

- **Description:** Optimize the application to handle high user traffic and large files by:
    - Integrating a content delivery network (CDN) for faster file delivery.
    - Implementing server-side caching to reduce redundant processing.
    - Introducing load balancing for backend servers.
- **Benefit:** Ensures reliable performance even under high demand.

7. Mobile-Friendly User Interface

- **Description:** Redesign the frontend to include responsive UI components for seamless access on mobile devices.
- **Benefit:** Expands accessibility for users who prefer mobile platforms for quick file conversions.

8. Multi-Language Support

- **Description:** Localize the user interface and error messages into multiple languages.
- **Benefit:** Makes the application accessible to a global audience.

9. Integration with Cloud Storage Services

- **Description:** Enable users to upload and download files directly from/to cloud storage platforms such as Google Drive, Dropbox, and OneDrive.
- **Benefit:** Adds convenience for users who work with cloud-based workflows.

10. Machine Learning for Content Analysis (Long-Term)

- **Description:** Incorporate AI models to analyze uploaded documents and provide insights, such as:
    - Extracting key phrases or summaries from documents.
    - Converting documents into accessible formats (e.g., audio or braille PDFs).

- **Benefit:** Provides value-added features for educational, professional, and accessibility needs.

11. Kubernetes Auto-Scaling

- **Description:** Enhance the existing Kubernetes setup by implementing auto-scaling to dynamically adjust the number of pods based on traffic.
- **Benefit:** Ensures efficient resource utilization and cost-effectiveness.

By implementing these enhancements, the application can evolve into a comprehensive document processing platform, offering users a feature-rich and highly reliable experience. This roadmap ensures adaptability to future demands while adhering to modern technological trends.

8 CONCLUSION

The Word-to-PDF Conversion Web Application successfully achieves its primary goal of converting .docx files into PDF format while providing a seamless user experience. By leveraging modern technologies such as React for the frontend, Node.js for the backend, and containerization with Docker, the application demonstrates a robust and scalable design. Additional deployment on platforms like Vercel and Render, coupled with CI/CD pipelines and Kubernetes orchestration, showcases adherence to industry best practices for modern web application development.

This project has been an excellent opportunity to combine software engineering principles with real-world application needs. The current implementation provides a foundation for further enhancements to expand functionality and usability.