### Project Description

The eBonding integration project aims to streamline the communication and data exchange between ServiceNow and external systems through automated processes. The project involves handling Word documents stored on OneDrive and utilizing Node.js for backend processing. The primary functionalities include:

1. \*\*Tag Extraction from Word Documents\*\*:

- Users upload a Word document to the system.

- The system extracts content control tags and their values from the document using XML parsing techniques.

- The extracted tags are returned as a JSON response.

2. \*\*Document Update with JSON Data\*\*:

- Users provide JSON data containing tag-value pairs.

- The system copies a Word template from OneDrive and creates a new document.

- The placeholders in the new document are replaced with the corresponding values from the JSON data.

- The updated document is available for download, and the temporary file is deleted post-download.

This project ensures that documents can be dynamically populated with data, reducing manual effort and improving efficiency in document management processes. The use of content control tags allows for precise identification and replacement of specific parts of the document, facilitating a seamless integration with various data sources.

### Objectives

1. \*\*Automate Document Processing\*\*:

- Develop a system to automate the extraction of tags and values from Word documents, reducing manual processing time and errors.

- Implement functionality to dynamically update Word documents with data from JSON inputs, ensuring up-to-date and accurate document content.

2. \*\*Streamline Data Integration\*\*:

- Integrate the system with ServiceNow to facilitate seamless data exchange between ServiceNow and external systems.

- Ensure the system can handle various types of data inputs and outputs, providing flexibility in data integration scenarios.

3. \*\*Enhance User Experience\*\*:

- Provide an intuitive and user-friendly interface for uploading documents and retrieving processed files.

- Ensure that the system is reliable and efficient, with minimal downtime and fast processing times.

4. \*\*Maintain Data Security\*\*:

- Implement robust security measures to protect sensitive data during upload, processing, and download of documents.

- Ensure compliance with relevant data protection regulations and best practices.

5. \*\*Facilitate Scalability\*\*:

- Design the system to handle increasing volumes of document processing requests as the user base grows.

- Ensure the architecture is modular and extensible to accommodate future enhancements and integrations.

6. \*\*Improve Operational Efficiency\*\*:

- Reduce the manual effort required for document processing, allowing staff to focus on higher-value tasks.

- Improve the accuracy and consistency of document content, leading to better decision-making and reduced operational risks.

### Technology Stack

1. \*\*Backend Framework\*\*:

- \*\*Node.js\*\*: Utilized for server-side scripting, allowing for efficient handling of asynchronous operations and I/O-bound tasks.

- \*\*Express.js\*\*: A minimal and flexible Node.js web application framework, used to build the RESTful API endpoints for document processing.

2. \*\*File Handling\*\*:

- \*\*Multer\*\*: Middleware for handling multipart/form-data, primarily used for uploading files.

- \*\*fs (File System)\*\*: Node.js module for interacting with the file system to read, write, and manipulate files.

3. \*\*Document Processing\*\*:

- \*\*JSZip\*\*: Library for reading and writing zip files, used to extract and update the contents of Word documents.

- \*\*xml2js\*\*: Library for converting XML to JavaScript objects and vice versa, facilitating the manipulation of Word document content.

- \*\*@xmldom/xmldom\*\*: DOM parser and XML serializer used for parsing and serializing XML documents.

- \*\*xpath\*\*: Library for querying and manipulating XML documents using XPath expressions.

- \*\*AdmZip\*\*: Library for creating, reading, and updating zip files.

4. \*\*UUID\*\*:

- \*\*uuid\*\*: Library for generating unique identifiers, used to create unique filenames for updated documents.

5. \*\*Logging\*\*:

- \*\*Console Logging\*\*: Standard console logging for debugging and tracking the flow of data and operations within the system.

6. \*\*Cloud Storage\*\*:

- \*\*OneDrive\*\*: Used to store the Word template files, allowing for easy access and management of templates.

7. \*\*Middleware\*\*:

- \*\*body-parser\*\*: Middleware for parsing incoming request bodies in a middleware, used to handle JSON payloads.

8. \*\*Development Tools\*\*:

- \*\*Visual Studio Code\*\*: Preferred IDE for development with support for various plugins and integrations.

- \*\*Postman\*\*: Tool for testing and documenting API endpoints, ensuring proper functionality and performance of the API services.

9. \*\*Version Control\*\*:

- \*\*Git\*\*: Version control system used to manage and track changes to the codebase, ensuring collaboration and version management.

### Learnings

1. \*\*Understanding of Document Processing\*\*:

- Gained deep insights into the structure of DOCX files and how to manipulate them programmatically using JavaScript libraries.

- Learned how to extract and replace content control tags within Word documents.

2. \*\*Proficiency in Node.js and Express.js\*\*:

- Enhanced understanding of building RESTful APIs using Node.js and Express.js.

- Learned to handle file uploads and manage file operations effectively in a server-side environment.

3. \*\*Working with XML and JSON\*\*:

- Improved skills in parsing and manipulating XML documents using libraries like `xml2js` and `@xmldom/xmldom`.

- Developed a better understanding of converting between XML and JSON formats.

4. \*\*Asynchronous Programming in JavaScript\*\*:

- Gained proficiency in handling asynchronous operations using promises and async/await in Node.js.

- Learned to manage and troubleshoot asynchronous workflows and file processing tasks.

5. \*\*Error Handling and Logging\*\*:

- Implemented robust error handling mechanisms to manage potential issues during file processing.

- Learned to use console logging for debugging and tracking the execution flow of the application.

6. \*\*Using Middleware for Enhanced Functionality\*\*:

- Learned to use middleware like `multer` for handling file uploads and `body-parser` for parsing incoming request bodies.

- Improved understanding of how middleware can enhance the functionality and security of a web application.

7. \*\*Integration with Cloud Storage Services\*\*:

- Gained experience in integrating with cloud storage services like OneDrive for storing and accessing template files.

- Learned to manage file operations and permissions in a cloud storage environment.

8. \*\*Version Control and Collaboration\*\*:

- Improved skills in using Git for version control, ensuring smooth collaboration and effective management of code changes.

- Learned to use branching, merging, and pull requests to manage project development effectively.

9. \*\*API Development and Testing\*\*:

- Gained experience in developing and testing API endpoints using tools like Postman.

- Learned to ensure API functionality and performance through thorough testing and documentation.

10. \*\*Project Management and Task Tracking\*\*:

- Enhanced understanding of using project management tools like JIRA to track tasks, bugs, and project progress.

- Learned to manage and prioritize tasks effectively, ensuring timely completion of project milestones.