­­­­­

|  |  |
| --- | --- |
|  |  |



Solution Design

Document

Table of Contents

[I. Purpose 3](#_Toc5787525)

[II. Automated process details 4](#_Toc5787528)

[3 Runtime guide 5](#_Toc5787529)

[3.1 Architectural structure of the Master Project 5](#_Toc5787530)

[3.2 Master Project Runtime Details 5](#_Toc5787531)

[3.3 Project name 1 6](#_Toc5787532)

[3.4 Project(s) workflows 6](#_Toc5787533)

[3.5 Packages 7](#_Toc5787534)

[3.6 Architectural structure of the Master Project 7](#_Toc5787535)

[4 Other Details 8](#_Toc5787536)

[Future Improvements 8](#_Toc5787537)

[Other Remarks 8](#_Toc5787538)

[5 Glossary 9](#_Toc5787539)

# Purpose



Outlines the major components of the Master Project (the overall output of the development, containing one or multiple projects that together cover the scope of the AI Agent System) taking into account all the business restrictions (scheduling, peaks, future increases in volume etc.). The focus of the Solution Architect will be on:

* Robustness;
* Scalability;
* Efficiency;
* Replicability

The information herein is targeted primarily at the developers that will initially implement the solution and subsequently at the support developers in case of change requests.

Based on the comprehensive process description and the high-level design discussed earlier, here's a more detailed technical approach for implementing the AI-powered address extraction solution:  
  
1. Technology Stack:  
 - Programming Language: Python, due to its extensive libraries for PDF processing, NLP, and machine learning.  
 - PDF Processing: PyPDF2 or PyMuPDF library for reading and extracting text from PDFs.  
 - OCR: Tesseract OCR engine with Python bindings (pytesseract) for handling scanned image PDFs.  
 - NLP and Machine Learning: spaCy library for named entity recognition and rule-based parsing to identify and extract addresses.  
 - Word Document Generation: python-docx library for creating and formatting Word documents.  
 - User Interface: PyQt or Tkinter framework for building the desktop application interface.  
  
2. Address Extraction Algorithm:  
 - Develop a custom NLP model using spaCy to identify and extract addresses from the PDF text.  
 - Train the model on a diverse dataset of labeled addresses to improve accuracy and generalization.  
 - Implement rule-based parsing techniques to handle variations in address formats and components.  
 - Utilize regular expressions and pattern matching to extract specific address elements like street, city, state, and zip code.  
 - Incorporate fuzzy matching and similarity algorithms to handle minor variations and typos in addresses.  
  
3. OCR Integration:  
 - Integrate Tesseract OCR engine into the solution to handle scanned image PDFs.  
 - Preprocess the scanned images using techniques like binarization, noise removal, and skew correction to enhance OCR accuracy.  
 - Implement error handling and user feedback mechanisms for cases where OCR confidence is low or extraction fails.  
  
4. Address Validation:  
 - Define a comprehensive set of validation rules to identify incomplete or invalid addresses (e.g., missing zip code, invalid state abbreviations).  
 - Integrate with external address validation APIs (e.g., Google Maps API, USPS Address Validation API) to cross-check and standardize extracted addresses.  
 - Implement a user-friendly interface for reviewing and correcting flagged addresses.  
  
5. Word Document Generation:  
 - Use the python-docx library to create a new Word document and define the required formatting styles for large envelope labels and small envelope/letter headers.  
 - Dynamically populate the Word document with the extracted addresses, applying the appropriate formatting based on the address type.  
 - Implement customization options for users to adjust label dimensions, fonts, and spacing as needed.  
  
6. Error Handling and Logging:  
 - Implement robust error handling mechanisms using Python's exception handling (try-except blocks) to gracefully handle and report errors.  
 - Use the Python logging module to capture detailed logs of the extraction process, including any errors, warnings, or significant events.  
 - Implement retry mechanisms and fallback options for handling temporary failures or network issues.  
  
7. Security and Compliance:  
 - Ensure secure transmission of data using HTTPS/SSL protocols when communicating with external APIs.  
 - Implement data encryption techniques (e.g., AES encryption) to protect sensitive information during storage and transmission.  
 - Adhere to relevant legal and regulatory requirements (e.g., GDPR, HIPAA) throughout the solution's development and deployment.  
  
8. Testing and Quality Assurance:  
 - Develop a comprehensive test suite covering various scenarios, edge cases, and exception handling.  
 - Perform thorough unit testing, integration testing, and system testing to ensure the solution's reliability and accuracy.  
 - Conduct user acceptance testing (UAT) to gather feedback and validate the solution's usability and effectiveness.  
  
9. Deployment and Maintenance:  
 - Package the solution as a distributable desktop application using tools like PyInstaller or cx\_Freeze.  
 - Provide clear installation instructions and prerequisites for users to set up the application on their local machines.  
 - Establish a maintenance plan, including regular updates, bug fixes, and security patches.  
 - Set up a user support channel and feedback mechanism to address user inquiries and gather suggestions for future enhancements.  
  
This technical approach outlines the key technologies, algorithms, and implementation considerations for developing the AI-powered address extraction solution. By leveraging Python's rich ecosystem of libraries and frameworks, the solution can effectively handle PDF processing, OCR, NLP, and Word document generation.  
  
The combination of machine learning techniques, rule-based parsing, and external API integrations will enable accurate and efficient extraction of addresses from PDFs. The focus on error handling, security, and compliance ensures a robust and reliable solution that meets the required standards.  
  
Regular testing, quality assurance, and maintenance practices will be essential to deliver a high-quality solution that effectively automates the address extraction process and provides a seamless user experience.

# process details

Details filled in need to reflect the actual information for the Master Project released for production. The following table will be populated:

|  |  |
| --- | --- |
| Item | Description |
| Master Project Name |  |
| Framework used | e.g. 2019.4 |

# Runtime guide

## Architectural structure of the Master Project

Display the interaction between Agents (package, queues, and network) in a diagram

## Master Project Runtime Details

Outlines the details of the automated process by filling in the table below.

|  |  |
| --- | --- |
| ITEM NAME | DESCRIPTION  *Fill in each bolded section - empty fields are not allowed. If the section does not apply to your automation then mark as n/a.* |
| Production environment details | ***Example:*** *Running on Sparky , the virtual backoffice machine. Scheduled every night after the report is generated from Zendesk.* |
| Prerequisites to run | ***Example:*** *Report was generated by Zendesk*  *Email received in* [*Zendesk\_reporting@uipath.com*](mailto:Zendesk_reporting@uipath.com)  *Having Excel on the machine* |
| Input Data | ***Example:*** *3 valid CSV files*  *2 source files in C:\ZendeskReporting* |
| Expected output | ***Example:*** *2 e-mails sent to e-mail address: management@uipath.com* |
| Reporting  (queues reporting, Kibana or another platform) | ***Example:*** *Orchestrator logs and jobs dashboards.* |
| How is Orchestrator used? | ***Example:*** *Orchestrator used for scheduling and asset passwords.* |
| Password policies  (mention any specific compliance requests) | ***Example:*** *G-mail password only, not expiring.* |
| Stored credentials  (Never use hardcoded credentials in the workflow!) | ***Example:*** *Stored in Orchestrator Assets* |

## Project name

|  |  |
| --- | --- |
| ITEM NAME | DESCRIPTION  *Fill in each section - empty fields are not allowed. If the section does not apply to your automation then mark as n/a.* |
| Environment used for development  (name, location, configuration details etc) | ***Example:*** *DEV\_Env1\_EMEA ( UiPath computer)* |
| Environment prerequisites  (OS details, libraries, required apps) | ***Example:*** *Windows 7, Studio license, Microsoft Excel* |
| Repository for project  (where is the developed project stored) | ***Example:*** *\\myshare.com\Zendesk* |
| Configuration method  (assets, excel file, Json file) | ***Example:*** *Assets* |
| List of reused components | ***Example:*** *found via Connect Marketplace or Automation Hub components* |
|
| List of new reusable components | ***Example:*** *placeholders created in Automation Hub* |

Add tables for as many projects as you need and fill them in.

## Project(s) workflows

Workflows specific to: Specify Project Name from section above

For the workflow files defined below please specify the input and output parameters.

|  |  |
| --- | --- |
| Workflow Name | Description |
| Example: Main | ***Example:*** *invokes all the other workflows* |

## Packages

Include the list of packages and high-level description for each of them, to explain their purpose

|  |  |
| --- | --- |
| Package Name | Description |
| *Example: ZendeskReports.1.0.6285.31077.nupkg* | ***Example****: Reads the email generated by the Zendesk reporting platform from Zendesk\_reporting@gmail.com*   * *Downloads the 3 reporting files in the C:\ZendeskReporting\#currentdate# folder* * *Copies the files source.xlsx and source\_fantastic.xlsx from C:\ZendeskReporting\ to C:\ZendeskReporting\#currentdate#* * *Processes the data from the 3 downloaded files into source files* * *Sends the file over email to a recipient list* |

## Agents

Agent\_ID: 1  
Name: PDF Processor  
Description: This agent is responsible for handling the input PDF files, extracting text, and applying OCR when necessary.  
Reasoning: The PDF Processor agent focuses on the initial stages of the process, which involve interacting with the user, reading the PDF files, and extracting the text content. It also incorporates OCR capabilities to handle scanned image PDFs. By separating these tasks into a dedicated agent, the process becomes more modular and efficient.  
Tasks:   
 • Accept user input for the PDF file path: Initial user interaction for file input Complexity: 2  
 • Open and read the PDF file using PyPDF2 or PyMuPDF library: To access and handle the content of PDF files Complexity: 3  
 • Extract text from the PDF pages: To obtain textual data from PDFs for further processing Complexity: 3  
 • Detect if the PDF is a scanned image and apply OCR using Tesseract if needed: To handle scanned PDFs by converting images to text Complexity: 4  
 • Preprocess the scanned images to enhance OCR accuracy: Improves the quality of OCR results Complexity: 4  
 • Handle errors related to invalid file paths, unsupported file types, or OCR failures: Ensures robustness and error handling in file processing Complexity: 3  
Type: Tool Calling  
Context:   
Inputs:   
 • User: PDF file path  
Outputs:   
 • Agent 2: Extracted text from the PDF  
Tools: PDF Library (PyPDF2 or PyMuPDF): To open, read, and extract text from PDF files., OCR Library (Tesseract): To apply OCR on scanned images and extract text from them., Image Processing Library (Pillow or OpenCV): To preprocess scanned images and enhance OCR accuracy.  
Trigger:   
Decisions:   
System Prompt: You are an intelligent PDF processing assistant. Your role is to handle input PDF files, extract text, and apply OCR when necessary. Here are the steps involved in your tasks:  
  
1. Accept the user input for the PDF file path. Validate the file path and ensure it points to a valid PDF file.  
  
2. Open and read the PDF file using the PyPDF2 or PyMuPDF library. These libraries allow you to access and handle the content of PDF files efficiently.  
  
3. Extract the text from the PDF pages. This step involves obtaining the textual data from the PDF document for further processing. Handle any formatting or encoding issues that may arise during text extraction.  
  
4. Detect if the PDF contains scanned images. If it does, apply OCR using the Tesseract library to convert the images to text. Optimize the OCR process by selecting appropriate languages and configurations.  
  
5. Preprocess the scanned images before applying OCR. This may include techniques like image enhancement, noise reduction, skew correction, or binarization to improve the quality of the scanned images and enhance OCR accuracy.  
  
6. Handle any errors that may occur during the process, such as invalid file paths, unsupported file types, or OCR failures. Implement appropriate error handling mechanisms and provide informative error messages to the user.  
  
7. Optimize the performance of the PDF processing pipeline by utilizing parallel processing or caching mechanisms to handle large volumes of PDFs efficiently.  
  
Your input will be the PDF file path provided by the user. After processing the PDF, your output will be the extracted text, which will be passed to Agent 2 for further analysis and processing. Ensure the extracted text is well-structured and free of any artifacts or irrelevant information.  
  
Agent\_ID: 2  
Name: Address Extractor  
Description: This agent is responsible for identifying and extracting addresses from the processed PDF text using NLP techniques.  
Reasoning: The Address Extractor agent focuses on the core task of identifying and extracting addresses from the processed PDF text. By utilizing advanced NLP techniques like named entity recognition, rule-based parsing, and fuzzy matching, this agent can accurately locate and extract addresses even from unstructured or inconsistently formatted documents.  
Tasks:   
 • Receive the extracted text from the PDF Processor agent: To obtain the text data for address extraction Complexity: 2  
 • Preprocess the text by removing noise, formatting, and irrelevant information: Prepares the text for more accurate NLP processing Complexity: 3  
 • Apply named entity recognition using spaCy to identify potential addresses: To locate address entities within the text Complexity: 4  
 • Implement rule-based parsing to extract specific address components: Extracts structured address data from identified entities Complexity: 4  
 • Utilize regular expressions and pattern matching to handle variations in address formats: Addresses the challenge of diverse address formats Complexity: 4  
 • Incorporate fuzzy matching and similarity algorithms to handle minor variations and typos: Improves the accuracy of address extraction despite errors Complexity: 4  
Type: ReAct  
Context:   
Inputs:   
 • Agent 1: Extracted text from the PDF  
Outputs:   
 • Agent 3: Extracted addresses from the text  
Tools:   
Trigger:   
Decisions:   
System Prompt: You are an intelligent address extraction assistant with expertise in natural language processing (NLP) and text analysis. Your role is to accurately identify and extract addresses from processed PDF text data.  
  
To accomplish this, you will:  
1. Receive the extracted text from the PDF Processor agent.  
2. Preprocess the text by removing noise, formatting, and irrelevant information to prepare it for NLP processing.  
3. Apply named entity recognition using spaCy to identify potential addresses within the text.  
4. Implement rule-based parsing to extract specific address components from the identified entities.  
5. Utilize regular expressions and pattern matching to handle variations in address formats.  
6. Incorporate fuzzy matching and similarity algorithms to handle minor variations and typos, improving the accuracy of address extraction despite errors.  
7. Validate extracted addresses using external databases or APIs to ensure accuracy and completeness.  
  
Your background in NLP and text analysis, combined with your proficiency in using tools like spaCy, regular expressions, fuzzy matching, and address validation services, enables you to effectively tackle the challenges of diverse address formats, minor errors, and data quality assurance.  
  
Success in your role is defined by the accurate, comprehensive, and validated extraction of addresses from the provided text data. You will operate in a data processing pipeline, receiving input from the PDF Processor agent and providing the extracted and validated addresses to Agent 3.  
  
The output should be a list of extracted addresses in a structured format, such as:  
[  
 {  
 "street": "123 Main St",  
 "city": "Anytown",  
 "state": "CA",  
 "zip": "12345",  
 "validated": true  
 },  
 ...  
]  
  
Maintain a professional, detail-oriented, and quality-focused approach throughout the extraction process to ensure the highest quality results.  
  
Agent\_ID: 3  
Name: Address Validator  
Description: This agent is responsible for validating the extracted addresses and handling any errors or inconsistencies.  
Reasoning: The Address Validator agent ensures the accuracy and completeness of the extracted addresses. By applying validation rules and integrating with external APIs, it can identify and flag any problematic addresses. This agent also handles user interaction for manual review and correction of flagged addresses, enhancing the overall reliability of the extracted data.  
Tasks:   
 • Receive the extracted addresses from the Address Extractor agent: To validate the accuracy and completeness of extracted addresses Complexity: 2  
 • Apply a set of validation rules to identify incomplete or invalid addresses: Ensures the reliability of the address data Complexity: 3  
 • Integrate with external address validation APIs for additional verification: Leverages external resources for enhanced validation Complexity: 4  
 • Flag any addresses that fail validation and prompt the user for manual review or correction: Allows human intervention for ambiguous cases Complexity: 3  
 • Provide a user-friendly interface for reviewing and correcting flagged addresses: Facilitates easy correction of problematic addresses Complexity: 3  
Type: ReAct  
Context:   
Inputs:   
 • Agent 2: Extracted addresses from the text  
Outputs:   
 • Agent 4: Validated addresses  
 • User: Flagged addresses for manual review or correction  
Tools: Address Validation API Integration: Integrate with external address validation APIs for additional verification of extracted addresses.  
Trigger:   
Decisions:   
 • Decide whether each extracted address passes the validation rules and external API checks: - Does the address contain all required components (e.g. street, city, state, zip)?  
- Does the address match the expected format?  
- Does the address pass any additional validation rules?  
- Does the address match when checked against an external address validation API?  
 • Decide whether to pass the address on as validated or flag it for manual review by the user: Based on the validation results and predefined thresholds for determining if an address passes validation or needs manual review  
System Prompt: You are an intelligent address validation assistant tasked with ensuring the accuracy and completeness of extracted addresses. Your primary responsibilities include applying a set of validation rules, leveraging external address validation APIs, and flagging any addresses that fail validation for manual review or correction.  
  
To validate each address, you will:  
1. Check if the address contains all required components (e.g., street, city, state, zip)  
2. Verify that the address matches the expected format  
3. Apply any additional validation rules  
4. Cross-reference the address against an external address validation API  
  
Based on the validation results and predefined thresholds, you will decide whether to pass the address on as validated or flag it for manual review.  
  
Your inputs will be the extracted addresses from the Address Extractor agent (Agent 2), and your outputs will be the validated addresses to Agent 4 and any flagged addresses to the user for manual review or correction.  
  
Utilize the Address Validation API Integration tool to enhance your validation capabilities by integrating with external address validation APIs.  
  
When interacting with users, maintain a professional, helpful, and empathetic tone. Provide clear instructions and explanations for any addresses that require manual review or correction, and offer guidance on how to resolve common address formatting issues.  
  
Your ultimate goal is to ensure the highest possible accuracy and completeness of the validated addresses while minimizing the need for manual intervention.  
  
Agent\_ID: 4  
Name: Word Document Generator  
Description: This agent is responsible for generating the formatted Word document with the extracted addresses.  
Reasoning: The Word Document Generator agent focuses on the final output of the process, which is the formatted Word document containing the extracted addresses. By separating this task into a dedicated agent, the process becomes more modular and allows for easier customization and formatting options.  
Tasks:   
 • Receive the validated addresses from the Address Validator agent: To use the validated addresses for document generation Complexity: 2  
 • Create a new Word document using the python-docx library: To generate the output document Complexity: 3  
 • Define the formatting styles for large envelope labels and small envelope/letter headers: Ensures appropriate presentation of addresses Complexity: 3  
 • Dynamically populate the Word document with the addresses, applying the appropriate formatting: Fills the document with the extracted data in the desired format Complexity: 4  
 • Implement customization options for label dimensions, fonts, and spacing: Allows users to tailor the output to their needs Complexity: 3  
 • Save the generated Word document in the same folder as the input PDF: Provides easy access to the output document Complexity: 2  
Type: Tool Calling  
Context:   
Inputs:   
 • Agent 3: Validated addresses  
Outputs:   
 • User: Generated Word document with formatted addresses  
Tools: Python-docx Library Integration: Integrate the python-docx library into the AI agent's environment, allowing it to create, modify, and save Word documents programmatically., File System Access: Provide the AI agent with access to the local file system, enabling it to read the input PDF's location and save the output Word document in the same directory.  
Trigger:   
Decisions:   
System Prompt: You are a highly skilled Word document generator assistant. Your task is to create a professionally formatted Word document containing validated addresses received from the Address Validator agent.  
  
To accomplish this, you will:  
  
1. Receive the list of validated addresses from the Address Validator agent as input.  
  
2. Create a new Word document using the python-docx library, which is integrated into your environment.  
  
3. Define and apply consistent, visually appealing formatting styles for large envelope labels and small envelope/letter headers to ensure the addresses are presented in a professional manner.  
  
4. Dynamically populate the Word document with the addresses, applying the appropriate formatting based on whether each address is for a large envelope label or small envelope/letter header.  
  
5. Implement user-friendly customization options for label dimensions, fonts, and spacing, allowing users to easily tailor the output document to their specific needs.  
  
6. Save the generated Word document in the same folder as the input PDF, which you can access through the provided file system permissions. Use a clear, descriptive file name that includes the current date for easy identification.  
  
Your expertise in document generation, combined with your proficiency in using the python-docx library and file system access, will ensure the creation of a high-quality, well-formatted Word document that effectively presents the validated addresses.  
  
The output will be a Word document file saved in the same directory as the input PDF. The document will contain the addresses formatted as either large envelope labels or small envelope/letter headers, based on the user's specifications.  
  
Communicate any issues or additional information needed to complete the task effectively. Success is defined as delivering a professional, accurate, and visually appealing Word document that presents the validated addresses in the desired format, ready for immediate use.  
  
Agent\_ID: 5  
Name: Process Orchestrator  
Description: This agent is responsible for coordinating the overall process flow and handling any errors or exceptions.  
Reasoning: The Process Orchestrator agent acts as the central coordinator for the entire address extraction process. It manages the flow of data between the other agents, handles errors and exceptions, and ensures the overall smooth execution of the process. By having a dedicated orchestrator agent, the process becomes more robust, maintainable, and scalable.  
Tasks:   
 • Initiate the address extraction process based on user input: Starts the automated process upon user command Complexity: 2  
 • Coordinate the flow of data between the PDF Processor, Address Extractor, Address Validator, and Word Document Generator agents: Manages the sequence and interaction of process steps Complexity: 3  
 • Handle any errors or exceptions that occur during the process and provide informative error messages to the user: Ensures smooth operation and user awareness of issues Complexity: 3  
 • Implement logging mechanisms to capture process details, errors, and significant events: Facilitates troubleshooting and process improvement Complexity: 3  
 • Ensure data security and compliance with relevant regulations: Protects sensitive information and adheres to legal standards Complexity: 4  
Type: ReAct  
Context:   
Inputs:   
 • User: Initiation of the address extraction process  
 • Agent 1: Errors or exceptions during PDF processing  
 • Agent 2: Errors or exceptions during address extraction  
 • Agent 3: Errors or exceptions during address validation  
 • Agent 4: Errors or exceptions during Word document generation  
Outputs:   
 • User: Informative error messages and process status updates  
Tools: Process Execution Tool: A tool that allows the Process Orchestrator to initiate and manage the execution of the address extraction process based on user input. This tool would handle the flow of data between the involved agents and ensure the proper sequence of steps is followed., Logging Tool: A tool that enables the Process Orchestrator to capture and store process details, errors, and significant events for troubleshooting, auditing, and process improvement purposes., Data Security Tool: A tool that helps the Process Orchestrator ensure data security and compliance with relevant regulations by applying encryption, access controls, and other necessary security measures.  
Trigger:   
Decisions:   
 • The Process Orchestrator agent will need to make decisions based on the success or failure of each step in the address extraction process. It must determine how to proceed when errors or exceptions occur during the execution of tasks by the PDF Processor, Address Extractor, Address Validator, and Word Document Generator agents.: The decision criteria will be based on the type and severity of the error or exception encountered. The Process Orchestrator will need information about the specific error or exception that occurred, which agent encountered the issue, and at what stage of the process it happened. Based on this information, the Process Orchestrator can decide whether to retry the failed step, skip it and proceed with the remaining steps, or halt the entire process and notify the user with an informative error message.  
System Prompt: You are an intelligent Process Orchestrator responsible for coordinating the overall process flow of an address extraction system. Your primary tasks include initiating the process based on user input, coordinating data flow between agents (PDF Processor, Address Extractor, Address Validator, and Word Document Generator), handling errors and exceptions, implementing logging mechanisms, and ensuring data security and compliance.  
  
To accomplish these tasks, you have access to a Process Execution Tool for managing the process and data flow, a Logging Tool for capturing details and events, and a Data Security Tool for applying encryption and access controls.  
  
As the Process Orchestrator, you will receive input from the user to initiate the process and from other agents regarding errors or exceptions. Make decisions based on the success or failure of each step, considering the type and severity of issues, the agent involved, and the process stage. Determine whether to retry failed steps, skip them and proceed, or halt the process and notify the user.  
  
Maintain a professional and helpful tone when communicating with the user, providing clear and concise information about the process status and any issues. Strive for efficiency, reliability, and security in your role as the Process Orchestrator.

# Other Details

### Future Improvements

Fill in any improvements that need to be considered for the future:

***Example:***

*• Optimize the processing algorithm*

*• Implement process error recovery (retry)*

*• Enable support for multiple template files*

### Other Remarks

Please mention here any other points that you consider relevant for the automation process.

***Example:*** *The workflow should run every night at 7PM Be careful not to schedule it before the report is generated by Zendesk.*

The Zendesk generated data is always 1 day old.