Solution Design Document (SDD)

LaTeX Document Translator - UiPath Automation

Document Information	
Document Title	Solution Design Document - LaTeX Translator
Project Name	TranslatorEvaluator
Project ID	84fd38a4-7f3d-40bd-90d0-2ae3b3a58cea
Version	1.0.0
Date	January 14, 2026
Author	Leustean Robert & Kurti Mark
Studio Version	UiPath Studio 26.0.182.0
Target Framework	Windows

Table of Contents

1. Solution Overview

1.1 Solution Summary

The TranslatorEvaluator is a UiPath automation solution designed to translate LaTeX documents while preserving their complex formatting structure. The solution uses a three-layer architecture (Presentation, Business, Data) with a supporting Framework layer for cross-cutting concerns.

1.2 Technology Stack

Component	Technology	Version
RPA Platform	UiPath	26.0.182.0
Expression Language	Visual Basic	-
Target Framework	Windows	-
Translation API	DeepL	v2
Configuration	Microsoft Excel	.xlsx
Caching	Text Files	.txt
Logging	Text Files	.txt

1.3 UiPath Dependencies

Package	Version	Purpose
UiPath.System.Activities	25.10.4	Core system activities
UiPath.Excel.Activities	3.3.1	Excel file operations
UiPath.WebAPI.Activities	2.3.2	HTTP/API operations
UiPath.FTP.Activities	3.0.0	File transfer (reserved)

1.4 Runtime Configuration

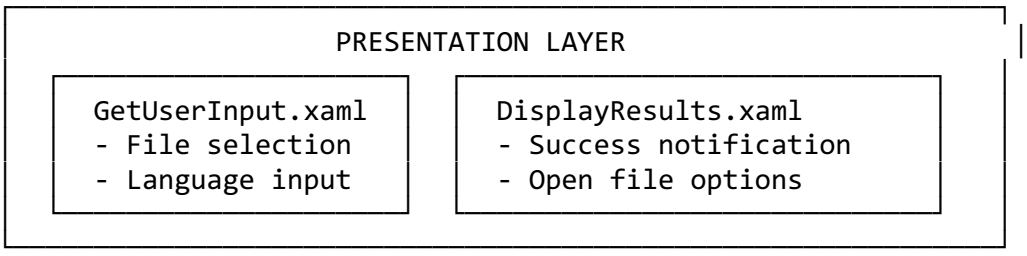
```
{
  "autoDispose": false,
  "netFrameworkLazyLoading": false,
  "isPausable": true,
  "isAttended": false,
  "requiresUserInteraction": true,
  "supportsPersistence": false,
  "executionType": "Workflow"
}
```

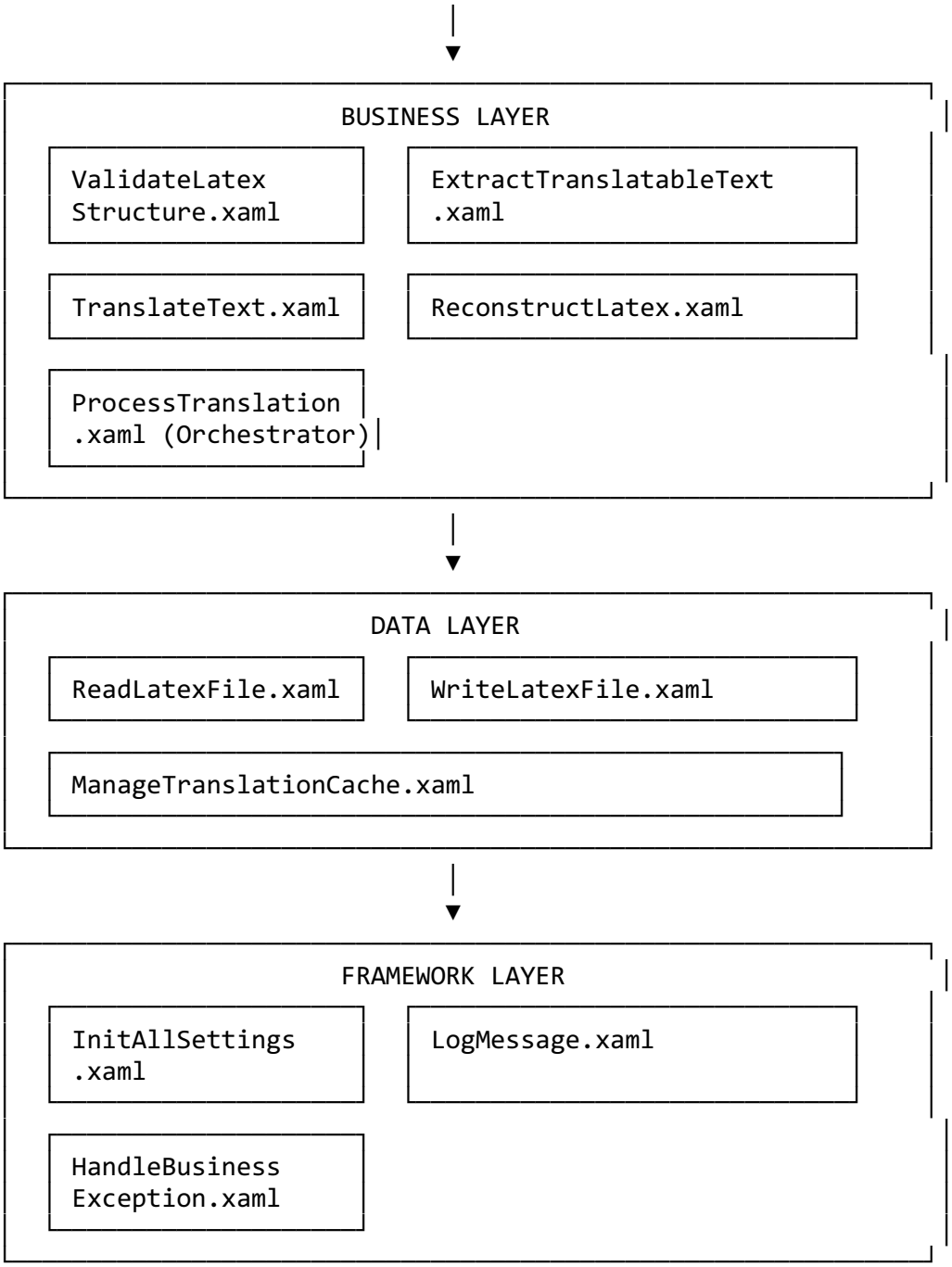
---

2. Architecture Design

2.1 Architectural Pattern

The solution follows a **Three-Layer Architecture** with separation of concerns:



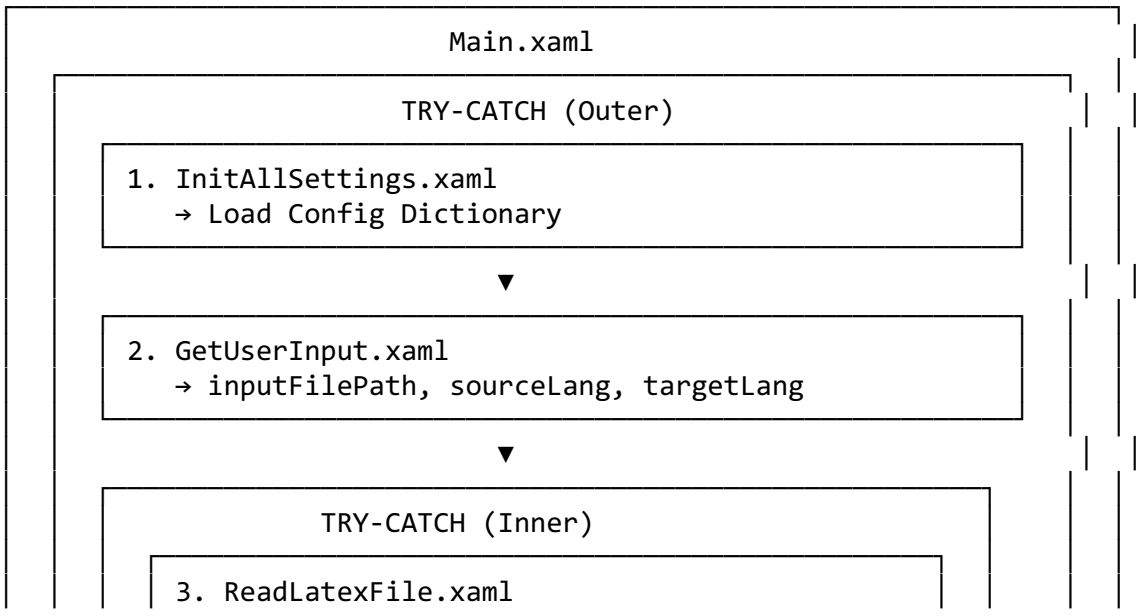


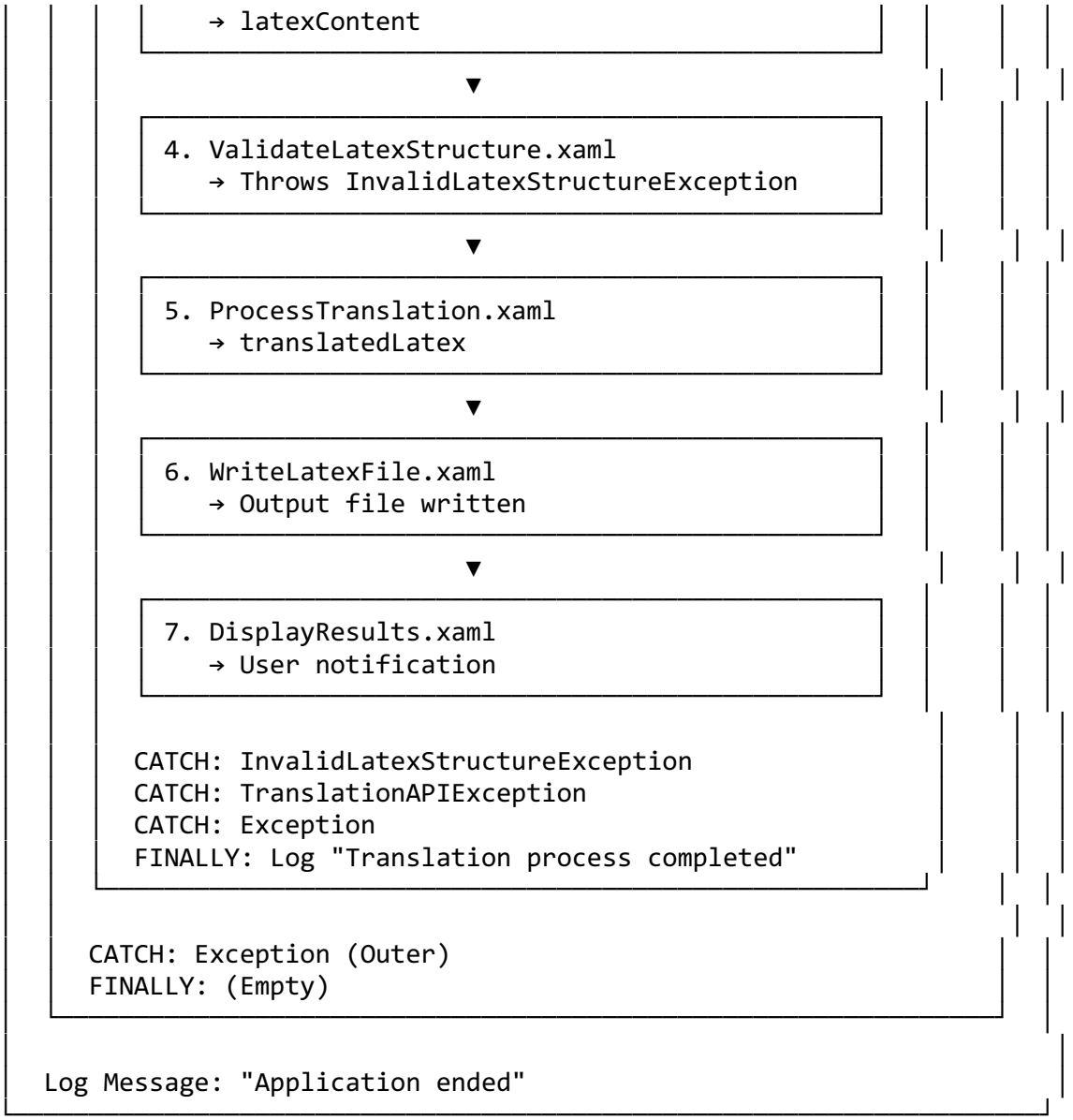
2.2 Folder Structure

```
TranslatorEvaluator/
├── Main.xaml                          # Entry point
├── project.json                       # Project configuration
│
├── BusinessLayer/                     # Business logic
│   ├── ExtractTranslatableText.xaml
│   ├── ProcessTranslation.xaml
```

```
│   ├── ReconstructLatex.xaml
│   ├── TranslateText.xaml
│   └── ValidateLatexStructure.xaml
│
├── DataLayer/                          # Data access
│   ├── ManageTranslationCache.xaml
│   ├── ReadLatexFile.xaml
│   └── WriteLatexFile.xaml
│
├── PresentationLayer/                  # User interface
│   ├── DisplayResults.xaml
│   └── GetUserInput.xaml
│
├── Framework/                          # Cross-cutting concerns
│   ├── HandleBusinessException.xaml
│   ├── InitAllSettings.xaml
│   └── LogMessage.xaml
│
├── Config/                             # Configuration files
│   └── Config.xlsx
│
├── Cache/                              # Translation cache
│   └── translations_{src}_{tgt}.txt
│
├── Logs/                               # Application logs
│   └── TranslationLog_YYYYMMDD.txt
│
├── InvalidLatexStructureException.cs   # Custom exception
└── TranslationAPIException.cs          # Custom exception
```

2.3 Execution Flow Diagram





---

3. Component Design

3.1 Main.xaml (Entry Point)

Purpose: Orchestrates the complete translation workflow

Variables: | Name | Type | Default | Description | |——|——|———|————-| | currentException | Exception | null | Current exception reference | | errorMessage | String | “” | Error message holder | | shouldRetry | Boolean | False | Retry flag | | translatedLatex | String | “” | Final translated content | | latexContent | String | “” | Original LaTeX content | | targetLang | String | “” | Target language code | | sourceLang | String | “” | Source language

code | | outputFilePath | String | “” | Output file path | | inputFilePath | String | “” | Input file path | | Config | Dictionary(String, Object) | new Dictionary | Configuration settings |

Output File Naming:

```
Path.Combine(
    Path.GetDirectoryName(inputFilePath),
    Path.GetFileNameWithoutExtension(inputFilePath) + "_translated_" + target
Lang + ".tex"
)
```

3.2 Framework Layer Components

3.2.1 InitAllSettings.xaml

Purpose: Loads configuration from Excel file

Arguments: | Direction | Name | Type | |————|——|——| | Out | out_Config | Dictionary(String, Object) |

Algorithm: 1. Set configPath = Path.Combine(CurrentDirectory, “Config”, “Config.xlsx”) 2. Initialize empty dictionary 3. Read “Settings” sheet into DataTable 4. For each row: Config(row(“Key”)) = row(“Value”) 5. Log “Configuration loaded: {count} settings”

3.2.2 LogMessage.xaml

Purpose: Writes timestamped log entries

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Config | Dictionary(String, Object) | | In | in_Level | String | | In | in_Message | String |

Log Entry Format:

```
[{yyyy-MM-dd HH:mm:ss}] [{Level}] {Message}
```

Log File Path:

```
Path.Combine(in_Config("LogFolder").ToString, "TranslationLog_" + DateTime.No
w.ToString("yyyyMMdd") + ".txt")
```

3.3 Presentation Layer Components

3.3.1 GetUserInput.xaml

Purpose: Collects user input for translation parameters

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Config | Dictionary(String, Object) | | Out | out_InputFilePath | String | | Out | out_SourceLang | String | | Out | out_TargetLang | String |

Input Validation Logic:

```
Do While NOT validInput
    InputDialog → tempPath
    tempPath = tempPath.Trim().Trim("""c)

    If File.Exists(tempPath) AND tempPath.EndsWith(".tex") Then
        out_InputFilePath = tempPath
        validInput = True
    Else
        MessageBox "Invalid file! Please provide a valid .tex file path."
    End If
Loop
```

Language Validation:

```
If out_SourceLang.Equals(out_TargetLang, StringComparison.OrdinalIgnoreCase)
Then
    Throw New ArgumentException("Source and target languages cannot be the sa
me")
End If
```

3.3.2 DisplayResults.xaml

Purpose: Shows completion notification and offers options

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Config | Dictionary(String, Object) | | In | in_OutputPath | String | | In | in_SourceLang | String | | In | in_TargetLang | String |

Features: - Displays translation summary - Option to open file in Notepad - Option to open output folder in Explorer

---

3.4 Business Layer Components

3.4.1 ValidateLatexStructure.xaml

Purpose: Validates LaTeX document structure

Arguments: | Direction | Name | Type | |————|——|——| | In | in_LatexContent | String |

Validation Checks:

Check	Condition	Exception
Document Class	Contains("\documentclass")	InvalidLatexStructureException("Missing \documentclass command")
Begin Document	Contains("\begin{document}")	InvalidLatexStructureException("Missing \begin{document}")
End Document	Contains("\end{document}")	InvalidLatexStructureException("Missing \end{document}")
Balanced Environments	beginCount = endCount	InvalidLatexStructureException("Unbalanced environments")

Environment Counting Algorithm:

```
beginCount = CInt((in_LatexContent.Length - in_LatexContent.Replace("\begin{",
 "").Length) / 7)
endCount = CInt((in_LatexContent.Length - in_LatexContent.Replace("\end{", "
").Length) / 5)
```

3.4.2 ExtractTranslatableText.xaml

Purpose: Separates translatable text from LaTeX structure

Arguments: | Direction | Name | Type | |————|——|——| | In | in_LatexContent | String | | Out | out_TranslatableSegments | List(String) | | Out | out_LatexTemplate | String |

Extraction Logic:

Protected Content (isInMathMode / isInEnvironment = True): - `$$...$$` (display math) - `\[...\]` (display math) - `\begin{equation}...\end{equation}` - `\begin{align}...\end{align}` - `\begin{figure}...\end{figure}` - `\begin{table}...\end{table}` - `\begin{lstlisting}...\end{lstlisting}`

Extracted Content: - `\section{...}` → Extract text between braces - `\subsection{...}` → Extract text between braces - `\subsubsection{...}` → Extract text between braces - `\caption{...}` → Extract text between braces - `\title{...}` → Extract text between braces - Plain text paragraphs (lines not starting with \)

Placeholder Format:

```
{{SEGMENT_0}}, {{SEGMENT_1}}, {{SEGMENT_2}}, ...
```

3.4.3 ProcessTranslation.xaml (Orchestrator)

Purpose: Coordinates the extraction, translation, and reconstruction

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Config | Dictionary(String, Object) | | In | in_LatexContent | String | | In | in_SourceLang | String | | In | in_TargetLang | String | | Out | out_TranslatedLatex | String |

Workflow: 1. Call ExtractTranslatableText → (latexTemplate, translatableSegments) 2. If segments.Count = 0 → Return original content 3. Initialize translatedSegments = New List(Of String) 4. For Each segment in translatableSegments: - Try: Call TranslateText → translatedSegment - Catch: Use original segment - Add to translatedSegments - Delay 1 second (rate limiting) 5. Call ReconstructLatex → out_TranslatedLatex 6. Append metadata comment

Metadata Footer:

```
% Translated by TranslatorEvaluator
% Source: {sourceLang} -> Target: {targetLang}
% Date: {yyyy-MM-dd HH:mm:ss}
% Segments translated: {count}
```

3.4.4 TranslateText.xaml

Purpose: Translates individual text segments

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Config | Dictionary(String, Object) | | In | in_Text | String | | In | in_SourceLang | String | | In | in_TargetLang | String | | Out | out_TranslatedText | String |

Cache Check Flow:

```
If useCache Then
    ManageTranslationCache(GET) → cacheFound, cachedTranslation
    If cacheFound Then
        out_TranslatedText = cachedTranslation
        Return
    End If
End If
```

DeepL API Call:

```
apiUrl = "https://api-free.deepl.com/v2/translate"
requestBody = "text=" + HttpUtility.UrlEncode(in_Text) +
              "&source_lang=" + in_SourceLang.ToUpper() +
              "&target_lang=" + in_TargetLang.ToUpper()

Using client As New HttpClient()
    client.DefaultRequestHeaders.Add("Authorization", "DeepL-Auth-Key " + api
Key.Trim())
    Dim content = New StringContent(requestBody, Encoding.UTF8, "application/
x-www-form-urlencoded")
```

```
    Dim response = client.PostAsync(apiUrl, content).Result
    responseText = response.Content.ReadAsStringAsync().Result
End Using
```

Response Parsing:

```
' DeepL returns: {"translations":[{"text":"translated text"}]}
If responseText.Contains("""text"":""") Then
    Dim startIdx = responseText.IndexOf("""text"":""") + 8
    Dim endIdx = responseText.IndexOf("""", startIdx)
    out_TranslatedText = responseText.Substring(startIdx, endIdx - startIdx)
    out_TranslatedText = out_TranslatedText.Replace("\\", "\")  ' Unescape ba
ckslashes
End If
```

3.4.5 ReconstructLatex.xaml

Purpose: Replaces placeholders with translated text

Arguments: | Direction | Name | Type | |————|——|——| | In | in_LatexTemplate | String | | In | in_TranslatedSegments | List(String) | | Out | out_ReconstructedLatex | String |

Algorithm:

```
out_ReconstructedLatex = in_LatexTemplate
segmentIndex = 0

For Each translatedText In in_TranslatedSegments
    placeholder = "{{SEGMENT_" + segmentIndex.ToString + "}}"
    out_ReconstructedLatex = out_ReconstructedLatex.Replace(placeholder, tran
slatedText)
    segmentIndex += 1
Next
```

3.5 Data Layer Components

3.5.1 ReadLatexFile.xaml

Purpose: Reads LaTeX source file

Arguments: | Direction | Name | Type | |————|——|——| | In | in_FilePath | String | | Out | out_Content | String |

Configuration: - Encoding: 65001 (UTF-8) - Error handling: IOException, Exception

---

3.5.2 WriteLatexFile.xaml

Purpose: Writes translated LaTeX file

Arguments: | Direction | Name | Type | |————|——|——| | In | in_FilePath | String | | In | in_Content | String |

Algorithm: 1. Get directory path 2. Create directory if not exists 3. Delete existing file if present 4. Append content (creates new file) 5. Log success message

---

3.5.3 ManageTranslationCache.xaml

Purpose: Manages persistent translation cache

Arguments: | Direction | Name | Type | |————|——|——| | In | in_Action | String | GET or SET | | In | in_SourceText | String | | In | in_SourceLang | String | | In | in_TargetLang | String | | In | in_TranslatedText | String | | Out | out_Found | Boolean | | Out | out_CachedTranslation | String |

Cache File Path:

```
Path.Combine(Environment.CurrentDirectory, "Cache", "translations_" + in_Sour
ceLang + "_" + in_TargetLang + ".txt")
```

Cache Key Generation (MD5 Hash):

```
Using md5 As MD5 = MD5.Create()
    Dim inputBytes As Byte() = Encoding.UTF8.GetBytes(in_SourceText)
    Dim hashBytes As Byte() = md5.ComputeHash(inputBytes)
    Dim sb As New StringBuilder()
    For Each b As Byte In hashBytes
        sb.Append(b.ToString("x2"))
    Next
    cacheKey = sb.ToString()
End Using
```

Cache File Format:

```
{md5hash}|{translated_text}
{md5hash}|{translated_text}
...
```

GET Action: - Load cache file into dictionary - Check if cacheKey exists - Return cached translation if found

SET Action: - Add/update entry in dictionary - Rebuild and save cache file

---

4. Data Design

4.1 Configuration Data (Config.xlsx)

Sheet: Settings

Key	Value	Description
DeepLAPIKey	{api_key}	DeepL authentication
DefaultSourceLang	en	Default source language
DefaultTargetLang	ro	Default target language
LogFolder	Logs	Log file directory
CacheTranslations	True	Enable caching
TranslationAPI	deepl	API provider

4.2 Cache Data Structure

File: `Cache/translations_{src}_{tgt}.txt`

Format: Pipe-delimited key-value pairs

```
{md5_hash_of_source}|{translated_text}
```

Example:

```
0b79795d3efc95b9976c7c5b933afce2|Introducere
29d5db0cf868ace1f9e7ab66c6824044|Acesta este un simplu articol de test pentru
 traducere.
```

4.3 Log Data Structure

File: `Logs/TranslationLog_YYYYMMDD.txt`

Format:

```
[{timestamp}] [{level}] {message}
```

Example:

```
[2026-01-14 17:19:50] [Info] User inputs - File: document.tex, From: en To: r
o
[2026-01-14 17:19:51] [Info] Processing translation from en to ro
[2026-01-14 17:20:52] [Info] Translation completed successfully
```

---

5. Integration Design

5.1 DeepL API Integration

Endpoint: `https://api-free.deepl.com/v2/translate`

Method: POST

Headers:

```
Authorization: DeepL-Auth-Key {api_key}
Content-Type: application/x-www-form-urlencoded
```

Request Body:

```
text={url_encoded_text}&source_lang={LANG}&target_lang={LANG}
```

Response Format:

```
{
  "translations": [
    {
      "detected_source_language": "EN",
      "text": "Translated text here"
    }
  ]
}
```

Error Handling: | HTTP Status | Meaning | Action | |————-|———|———| | 200 | Success | Parse and use translation | | 400 | Bad Request | Log error, use original | | 401 | Unauthorized | Show API key error | | 403 | Forbidden | Show quota error | | 429 | Too Many Requests | Retry after delay | | 500+ | Server Error | Use original text |

5.2 File System Integration

Operation	Activity	Notes
Read Text	ReadTextFile	UTF-8 encoding
Write Text	AppendLine	Creates if not exists
Check Exists	File.Exists	VB expression
Create Directory	Directory.CreateDirectory	InvokeCode
Delete File	File.Delete	InvokeCode

---

6. Error Handling Design

6.1 Custom Exceptions

InvalidLatexStructureException

```
namespace TranslatorEvaluator
{
    public class InvalidLatexStructureException : Exception
    {
        public string LatexSection { get; set; }
        public int LineNumber { get; set; }

        public InvalidLatexStructureException(string message, string section,
 int line)
            : base(message)
        {
            LatexSection = section;
            LineNumber = line;
        }
    }
}
```

Usage:

```
Throw New TranslatorEvaluator.InvalidLatexStructureException(
    "Missing \documentclass command",
    "Document Header",
    0
)
```

TranslationAPIException

```
namespace TranslatorEvaluator
{
    public class TranslationAPIException : Exception
    {
        public int StatusCode { get; set; }
        public string APIResponse { get; set; }

        public TranslationAPIException(string message, int statusCode, string
 response)
            : base(message)
        {
            StatusCode = statusCode;
            APIResponse = response;
        }
    }
}
```

6.2 Exception Handling Strategy

Main.xaml Exception Hierarchy:

```
TryCatch (Outer)
├── Try:
│   ├── InitAllSettings
│   ├── GetUserInput
│   └── TryCatch (Inner)
│       ├── Try:
│       │   ├── ReadLatexFile
│       │   ├── ValidateLatexStructure
│       │   ├── ProcessTranslation
│       │   ├── WriteLatexFile
│       │   └── DisplayResults
│       ├── Catch InvalidLatexStructureException:
│       │   └── MessageBox("LaTeX Structure Error: " + exception.Message)
│       ├── Catch TranslationAPIException:
│       │   └── MessageBox("Translation API Error: " + exception.Message)
│       ├── Catch Exception:
│       │   └── MessageBox("Unexpected error: " + exception.Message)
│       └── Finally:
│           └── Log("Translation process completed")
├── Catch Exception:
│   └── (Empty - handled by inner catch)
└── Finally:
    └── (Empty)
```

---

7. Security Design

7.1 Sensitive Data Handling

Data Type	Protection Method
API Key	Stored in Config.xlsx, not logged
File Paths	Logged without content
Translations	Cached locally

7.2 Logged Data Exclusions

From `project.json`:

```
"excludedLoggedData": [
  "Private:*",
  "*password*"
]
```

7.3 Best Practices

1. **API Key Storage:** Store in UiPath Orchestrator Assets for production
2. **Cache Security:** Cache files contain translated text only, no source mapping
3. **Log Sanitization:** No sensitive content logged
4. **Network:** HTTPS only for API calls

---

8. Testing Strategy

8.1 Unit Test Cases

ID	Component	Test Case	Expected Result
TC-01	ValidateLatexStructure	Valid document	No exception
TC-02	ValidateLatexStructure	Missing \documentclass	InvalidLatexStructureException
TC-03	ValidateLatexStructure	Unbalanced environments	InvalidLatexStructureException
TC-04	ExtractTranslatableText	Document with math	Math not in segments
TC-05	ExtractTranslatableText	Section titles	Titles extracted
TC-06	TranslateText	Valid text	Translated text returned
TC-07	TranslateText	Cache hit	Cached value returned
TC-08	ReconstructLatex	All placeholders	All replaced
TC-09	ManageTranslationCache	SET then GET	Same value returned
TC-10	ReadLatexFile	Non-existent file	IOException

8.2 Integration Test Cases

ID	Test Case	Steps	Expected Result
IT-01	End-to-end translation	Full workflow	Translated file created
IT-02	API failure recovery	Mock API failure	Original text preserved
IT-03	Cache functionality	Translate same text twice	Second uses cache

8.3 Test Data

Sample Valid LaTeX:

```latex
\documentclass{article}
\begin{document}
\title{Test Document}
\section{Introduction}
This is sample text.
$$E=mc^2$$
\section{Conclusion}
Final text.
\end{document}
```

Sample Invalid LaTeX:

```latex
\begin{document}
No document class!
\end{document}
```

---

9. Deployment Guide

9.1 Prerequisites

Requirement	Details
UiPath Robot	Version 26.0+
.NET Framework	As required by UiPath
Network Access	api-free.deepl.com:443
DeepL Account	Free or Pro API key

9.2 Deployment Steps

1. **Package the Process:**

   ```
   UiPath Studio → Publish → Create Package
   ```

2. **Deploy to Orchestrator (Optional):**
   - Upload package to Orchestrator
   - Create Process
   - Assign to Robot
3. **Configure Settings:**
   - Copy Config.xlsx template
   - Add DeepL API key
   - Set default languages
4. **Create Folders:**
   - Ensure Cache/ folder exists
   - Ensure Logs/ folder exists

9.3 Configuration Checklist

- ☐ DeepLAPIKey set in Config.xlsx
- ☐ DefaultSourceLang configured
- ☐ DefaultTargetLang configured
- ☐ LogFolder path valid
- ☐ Network access to DeepL API verified

---

10. Maintenance Guide

10.1 Log Management

Daily Log Files: `Logs/TranslationLog_YYYYMMDD.txt`

Recommended Actions: - Archive logs older than 30 days - Monitor for Error level entries - Review Warn level for API issues

10.2 Cache Management

Cache Files: `Cache/translations_{src}_{tgt}.txt`

Recommended Actions: - Clear cache if translations seem stale - Monitor cache file sizes (large files may slow loading) - Backup before clearing

10.3 Common Issues

Issue	Cause	Resolution
API Key Error	Invalid or expired key	Update Config.xlsx
File Not Found	Incorrect path	Verify .tex file exists
Unbalanced Environments	LaTeX syntax error	Fix source document
Empty Translation	API quota exceeded	Check DeepL account

10.4 Version History

Version	Date	Changes
1.0.0	2026-01-14	Initial release

---

11. Appendices

Appendix A: Variable Reference

Main.xaml Variables:

Variable	Type	Scope	Purpose
Config	Dictionary(String, Object)	Main	Configuration store

Variable	Type	Scope	Purpose
inputFilePath	String	Main	User-selected file
outputFilePath	String	Main	Generated output path
latexContent	String	Main	Original file content
translatedLatex	String	Main	Translated content
sourceLang	String	Main	Source language code
targetLang	String	Main	Target language code

Appendix B: Activity Reference

Activity	Package	Usage
ReadTextFile	UiPath.System.Activities	File reading
AppendLine	UiPath.System.Activities	File writing
ReadRange	UiPath.Excel.Activities	Excel reading
InputDialog	UiPath.System.Activities	User input
MessageBox	UiPath.System.Activities	Notifications
LogMessage	UiPath.System.Activities	Logging
InvokeCode	UiPath.System.Activities	VB.NET code
InvokeWorkflowFile	UiPath.System.Activities	Sub-workflow calls

Appendix C: API Reference

DeepL Free API: - Base URL: `https://api-free.deepl.com` - Rate Limit: Based on account tier - Character Limit: 500,000/month (free tier)

Supported Language Codes: | Code | Language | |——|————-| | EN | English | | DE | German | | FR | French | | ES | Spanish | | PT | Portuguese | | IT | Italian | | NL | Dutch | | PL | Polish | | RU | Russian | | JA | Japanese | | ZH | Chinese | | RO | Romanian |
