Part 1: Functional Testing (Manual)

1. Test a document translation feature by uploading the following types of files:

○ Text documents (e.g., .docx, .txt, .pdf).

○ Media files with embedded text (e.g., images with text).

2. Verify translations for accuracy, file integrity, and formatting preservation after translation. Deliverables:

● Provide test cases for uploading, translating, and downloading the translated document.

Testcases ID	Functionalities	Test Case Descriptions	Steps to Reproduce	Expected Results
TC-01.01	**Upload**	Verify uploading supported text document formats (.docx, .txt).	1. Navigate to the upload page. 2. Select a valid .docx or .txt file. 3. Click on "Upload".	File should be uploaded successfully with a confirmation message.
TC-01.02		Verify uploading a supported PDF document.	1. Navigate to the upload page. 2. Select a valid .pdf file. 3. Click on "Upload".	File should be uploaded successfully with a confirmation message.
TC-01.03		Verify uploading unsupported file types.	1. Navigate to the upload page. 2. Select an unsupported file type (e.g., .exe, .mp4). 3. Click on "Upload".	Error message should be displayed: "Unsupported file type."

TC-01.04		Verify uploading media files with embedded text (e.g., images).	1. Navigate to the upload page. 2. Select an image file (e.g., .jpg with embedded text). 3. Click on "Upload".	File should be uploaded successfully, with OCR applied to extract text for translation.
TC-01.05		Verify max file size limit.	1. Navigate to the upload page. 2. Select a file exceeding the max size limit. 3. Click on "Upload".	Error message should be displayed: "File exceeds size limit."
TC-02.01	**Translation**	Verify translation accuracy for text documents.	1. Upload a .docx or .txt file. 2. Initiate translation to a target language. 3. Verify output text against known correct translations.	Translated text should be accurate.
TC-02.02		Verify formatting preservation for .docx files.	1. Upload a .docx file with varied formatting (e.g., bold, italic, tables). 2. Translate the file.	Formatting should be preserved in the translated document.

			3. Check the formatting in the output.	
TC-02.03		Verify text extraction accuracy from media files.	1. Upload an image with embedded text. 2. Initiate translation. 3. Check extracted and translated text.	Extracted text should be matched with the original text; translation should be accurate.
TC-02.04		Verify translation process for large documents.	1. Upload a large document with several pages. 2. Initiate translation. 3. Wait for completion.	Translation should be completed successfully within the expected time.
TC-02.05		Verify translations with complex language structures.	1. Upload a document with idioms, jargon, or complex sentences. 2. Translate to target language. 3. Verify accuracy.	Translations should be contextually and grammatically accurate.
TC-03.01	**Download**	Verify downloading translated text documents.	1. Translate a .docx or .txt file.	File should be downloaded successfully in the chosen format.

			2. Click "Download".	
TC-03.02		Verify downloading translated PDF documents.	1. Translate a .pdf file. 2. Click "Download".	File should be downloaded successfully; formatting and content are preserved.
TC-03.03		Verify the integrity of downloaded files.	1. Translate and download any document. 2. Open the file. 3. Verify content and formatting.	Content should be matched with the translation preview; no corruption observed.
TC-03.04		Verify retry mechanism for failed downloads.	1. Simulate a network failure during download. 2. Retry downloading the translated document.	Retry functionality should work properly; file should be downloaded successfully.

● Report any issues found, categorized by severity.

Bugs ID	**Bug Summary**	**Steps to Produce**	**Expected Results**	**Actual Results**	**Severity**
01	File upload crashes for large files.	Upload file > max size limit.	Error messages should be displayed.	Application crashes.	Critical

02	Formatting lost in .docx translations.	Translate a formatted .docx file.	Formatting should be preserved.	Formatting is inconsistent.	High
03	Minor errors in OCR text extraction.	Upload image with text. Translate.	Text extraction should be accurate.	Text extraction has errors.	Medium
04	UI message typos during translation.	Trigger translation. View messages.	Messages should be grammatically correct.	Typos in system messages.	Low

Part 2: Automated Testing

Task: Write an automated script to test the file upload and download functionality of the document translation product.

Requirements:

● Use a framework or tool of your choice (e.g., Selenium, Python unittest, or Postman for API).

● Ensure the script verifies file integrity post-download.

Deliverables:

● Submit the script along with a brief explanation of your approach.

Functionalities	API Endpoints	Steps	Postman Requests

File Upload	api/upload	1. **Send a POST request** to upload a file.
		2. Verify that the response status is 200 (or the
			• **Method**: POST
			• **URL**: https://example.com/api/upload
			• **Headers**:
			{
			"Authorization": "Bearer {{auth_token}}",
			"Content-Type": "multipart/form-data"
			}

		expected status code).
		3. Confirm that the response contains a unique file ID or reference.
			• **Body**: Select "form-data" and upload a file under the file key.
			Tests Script:
			pm.test("Response status is 200", () => {
			  pm.response.to.have.status(200);
			});
			
			pm.test("Response contains file ID", () => {
			  const response = pm.response.json();
			  pm.expect(response).to.haveOwnProperty("fileId");
			pm.collectionVariables.set("fileId", response.fileId);
			});

File Translation	api/translate	1. **Send a POST request** to initiate translation using the uploaded file's ID.
		2. Verify that the response contains a translationId and that the translation is queued or in progress.
			Method: POST
			URL: https://example.com/api/translate
			Headers:
			{
			 "Authorization": "Bearer {{auth_token}}",
			 "Content-Type": "application/json"
			}
			Body:
			{
			 "fileId": "{{fileId}}",
			"targetLanguage": "FRANCE"
			}
			Tests Script:
			pm.test("Response status is 202", () => {
			  pm.response.to.have.status(202);
			});

			pm.test("Response contains translation ID", () =>
			{
			const response = pm.response.json();
			
			pm.expect(response).to.haveOwnProperty ("translationId");
			pm.collectionVariables.set("translationId", response.translationId);
			});

File Download	api/download	1. **Send a GET request** to download the translated file using the translationId.
		2. Verify the response status is 200 and that the content type matches the file format.
		3. Confirm file integrity by checking content size or hash.
			• **Method**: GET
			• **URL**: https://example.com/api/download?translationId={{translationId}}
			• **Headers**:
			{
			  "Authorization": "Bearer {{auth_token}}"
			}
			Tests Script:
			pm.test("Response status is 200", () => {
			   pm.response.to.have.status(200);
			});
			
			pm.test("Response contains file content", () =>
			{
			   const contentDisposition = pm.response.headers.get("Content-Disposition");
			   pm.expect(contentDisposition).to.include("attachment; filename=");
			});
			
			pm.test("Response content type is valid", () =>
			{
			   const contentType = pm.response.headers.get("Content-Type");

			pm.expect(contentType).to.match(/application\/(pdf|msword|octet-stream)/);
			});

Brief Explanation

- **File Upload Test**:
  1. Verifies the API's ability to accept file uploads.
  2. Ensures the API returns a unique file ID for further operations.
- **File Translation Test**:
  1. Initiates translation for the uploaded file.
  2. Confirms that the translation request is successfully accepted and a translationId is generated.
- **File Download Test**:
  1. Tests the API's ability to provide the translated file for download.
  2. Verifies file integrity by checking headers and content type.

Part 3: Performance Testing

Task: Evaluate the system's response time and scalability for the document translation feature:

1. Upload and translate documents of varying sizes (small: 1MB, medium: 10MB, large:
100MB).

2. Test concurrent user uploads using a load testing tool (e.g., JMeter).

Deliverables:

● Report on response times and any performance bottlenecks.

Functionalities	API Endpoints	Steps	Postman Requests
File Upload	api/upload	**1.** Send a POST request **2.** Use HTTP Request Samplers **3.** Prepare three sample files: • Small: 1MB text document.	**1. Thread Group:** Set the number of threads (users), ramp-up period, and loop count. • Threads: 50/100/500/1000

		• Medium: 10MB text document. • Large: 100MB text document. **4.** Measure response times for: • Uploading the file. **5.** Record results and compare them against SLAs.	• Ramp-up Period: 10 seconds • Loop Count: 1 • Add Timers: Use a Constant Timer or Uniform Random Timer to simulate real-world delays. • Listeners: - Add listeners for result analysis: - Aggregate Report - Summary Report - View Results in Table - Response Time Graph **2. HTTP Request (Upload):** • **Method:** POST • **URL:** https://example.com/api/upload • **Body Data:** Use the **File Upload** feature in JMeter. • **Add multipart/form-data header:** Content-Type: multipart/form-data **3. Add CSV Data Set Config:** • Use a CSV file to parameterize the file paths (e.g., small.txt, medium.txt, large.txt). **4. Add Assertions:** • Response Code = 200 • Response Body contains fileId. **5. Save Responses:** • Save the response file ID for the next step using a Regular Expression Extractor.
File Translation	api/translate	1. Send a POST request 2. Use HTTP Request Samplers	**1. Thread Group:**

		3. Prepare sample file
		• Uploading the file.
		• Record results and compare them against SLAs.
			Set the number of threads (users), ramp-up period, and loop count.
			• Threads: 50/100/500/1000
			• Ramp-up Period: 10 seconds
			• Loop Count: 1
			• Add Timers: Use a Constant Timer or Uniform Random Timer to simulate real-world delays.
			• Listeners:
			- Add listeners for result analysis:
			- Aggregate Report
			- Summary Report
			- View Results in Table
			- Response Time Graph
			2. HTTP Request (Upload):
			• **Method:** POST
			• **URL:** https://example.com/api/translate
			• **Body Data:**
			{
			"fileId": "${fileId}", "targetLanguage": "FRANCE"
			}
			• **Add header:**
			- Content-Type: application/json
			- Authorization: Bearer YOUR_TOKEN
			3. Add Assertions:
			• Response Code = 202

			• Response Body contains translationId. 4. **Save Responses:** Extract the translationId using a Regular Expression Extractor.
File Download	api/download	1. Send a GET request 2. Use HTTP Request Samplers 3. Prepare sample file 4. Uploading the file. 5. Record results and compare them against SLAs.	1. **Thread Group:** Set the number of threads (users), ramp-up period, and loop count. • Threads: 50/100/500/1000 • Ramp-up Period: 10 seconds • Loop Count: 1 • Add Timers: Use a Constant Timer or Uniform Random Timer to simulate real-world delays. • Listeners: - Add listeners for result analysis: - Aggregate Report - Summary Report - View Results in Table - Response Time Graph 2. **HTTP Request (Upload):** • **Method:** GET • **URL:** https://example.com/api/translate **3. Add Assertions:** Response Code = 200 Response Header contains Content-Disposition: attachment. 4. **Save Responses:**

			Use the **Save Responses to a File** listener to save the downloaded file.

Part 4: End-to-End Testing

Task:

Design and execute an end-to-end test for the following scenario:

1. A user uploads a document for translation.
2. The document is translated and downloaded.
3. The translated document is shared via a social network AI agent.

Deliverables:

● Provide a detailed test case and execution report for the scenario, including screenshots or logs.

#TC	Test Scenarios	Testcases	Preconditions	Test Data	Steps to Reproduce	Expected Results
01	Upload a document for translation.					
02	Download the translated document successfully.					
03	Share the translated document via a social network AI agent.					
03.01		Document Translation and Sharing	1. The user is logged in to the application.	A sample document: sample_text.docx	1. Navigate to the upload page. 2. Upload sample_text.docx.	1. Translated file content matches the expected output.

			2. Social network AI agent is configured.		Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion. 4. Download the translated document and verify its format and content. 5. Open the "Share" functionality. Select the social network AI agent as the sharing method. 6. Confirm the sharing action and verify the AI agent successfully shares the translated document to	2. Sharing via the AI agent is successful, and confirmation is received on the social network.

EXCUTION REPORTS

#TC	Test Scenarios	Step Description	Test Environments	Expected Results	Status	Logs/Screenshots	#Defect
01.01	Upload a document for translation.	1. Navigate to the upload page. 2. Upload sample_text.docx	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88	Document upload is successful, and a confirmation message is displayed.	Passed	Document upload page loaded successfully. Screenshot: upload_page.png.	
01.02	Upload a document for translation.	1. Navigate to the upload page. 2. Upload sample_text.docx	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88	Upload sample_text.docx. Verify the document is successfully uploaded.	Passed	File sample_text.docx uploaded successfully. Confirmation message: "Upload successful." Screenshot: upload_success.png.	
02.01	Download the translated document successfully.	1. Navigate to the upload page. 2. Upload sample_text.docx. Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion.	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88	Translation completes without errors.	Passed	Translation completed. Log: Translation completed in 12 seconds. Screenshot: translation_success.png.	

02.02	Download the translated document successfully.	1. Navigate to the upload page. 2. Upload sample_text.docx. Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion. 4. Download the translated document and verify its format and content.	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88	The translated file is downloadable.	Passed	File downloaded successfully. File name: sample_text_translated.docx. Content verified. Screenshot: file_downloaded.png.	
02.03	Download the translated document successfully.	1. Navigate to the upload page. 2. Upload sample_text.docx. Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion. 4. Download the translated	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88	Translated file content matches the expected output.	Passed	File downloaded successfully. File name: sample_text_translated.docx. Content verified. Screenshot: file_downloaded.png.	

		document and verify its format and content.					
03.01	Share the translated document via a social network AI agent.	1. Navigate to the upload page. 2. Upload sample_text.docx. Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion. 4. Download the translated document and verify its format and content. 5. Open the "Share" functionality. Select the social network AI agent as the sharing method.	**Operating System:** Windows 11 **Browser:** Chrome Version 118.0.5993.88 **Social Network AI Agent:** Facebook Messenger API	Sharing via the AI agent is successful.	Passed	Sharing UI loaded. Selected "Facebook Messenger AI Agent." Screenshot: sharing_ui.png.	
03.02	Confirm the sharing action and verify the AI	1. Navigate to the upload page. 2. Upload sample_text.docx	**Operating System:** Windows 11	Sharing via the AI agent is successful, and confirmation is	Passed	Document shared successfully. Confirmation message received on Messenger:	

	agent successfully shares the translated document to the specified social network.	. Verify the document is successfully uploaded. 3. Trigger the translation process and wait for completion. 4. Download the translated document and verify its format and content. 5. Open the "Share" functionality. Select the social network AI agent as the sharing method. 6. Confirm the sharing action and verify the AI agent successfully shares the translated document to the specified social network.	**Browser:** Chrome Version 118.0.5993.88 **Social Network AI Agent:** Facebook Messenger API	received on the social network.		"Document shared successfully." Screenshot: sharing_success.png.	

Part 5: Analytical and Creative Thinking

Task:

Identify three potential edge cases or unusual scenarios for each product (document/media translation, Gen AI, AI agents).

Deliverables:

● Provide the edge cases and suggest how you would test them.

Edge Cases and Testing

1. Document/Media Translation

1. **Mixed Languages:** Document contains multiple languages.
   - Test: Validate detection and translation accuracy for each language.
2. **Embedded Content:** File includes images or charts with text.
   - Test: Confirm proper OCR handling and layout preservation.
3. **Oversized Files:** File exceeds size limit.
   - Test: Check error messages and system stability.

2. Generative AI (Gen AI)

1. **Ambiguous Prompts:** Input is vague or philosophical.
   - Test: Assess response clarity and logical consistency.
2. **Conflicting Instructions:** Input contains contradictory requirements.
   - Test: Verify prioritization or request for clarification.
3. **Sensitive Topics:** Prompt involves controversial subjects.
   - Test: Ensure ethical handling and refusal of harmful outputs.

3. AI Agents

1. **Multilingual and Informal Input:** Mixed languages and slang.
   - Test: Validate accurate understanding and response.
2. **Conflicting Commands:** Rapidly toggled contradictory instructions.
   - Test: Check stability and prioritize the latest input.
3. **Vague Requests:** Incomplete or unclear commands.
   - Test: Confirm clarification queries or fallback mechanisms.