Freelance Project Scope: Automated PDF Merge and Email Distribution

Project Overview: Develop a standalone Python script that automatically merges daily incoming PDF files and emails the combined PDF to designated stakeholders.

Deliverables

Core Functionality

- **Automated PDF Merging**: Script that monitors a designated folder/location for incoming PDF files and merges them into a single combined PDF
- **Email Distribution**: Automatically send the merged PDF to a configurable list of stakeholders
- **Scheduling Capability**: Daily automated execution at specified time(s)
- **File Management**: Organize processed files (move to archive folder or maintain processing history)

Configuration Requirements

- **Flexible Input Sources**: Support monitoring local folders or network drives for incoming PDFs
- **Configurable Recipients**: Easy-to-update stakeholder email list without code modification
- **Customizable Settings**:
    - Email subject and body text
    - Merge frequency (daily timing)
    - File naming conventions for merged PDFs
    - Source and destination folder paths
- **Error Handling**: Logging and notification for failures (missing files, merge errors, email delivery issues)

Implementation Requirements

- **Modular Design: Separate components for:**
    - **PDF collection and validation**
    - **PDF merging operations**
    - **Email sending functionality**
    - **Scheduling and automation**
    - **Logging and error handling**
- **Configuration File: All parameters (email credentials, folder paths, recipients, timing) managed through configuration file**
- **Documentation: Clear setup and usage instructions**
- **Duplicate Detection: Identify and skip duplicate files based on filename**

Deliverables Package

1. Complete Python script(s) organized by functionality
2. Dependency specifications (requirements file)
3. Configuration file template
4. Documentation including:
    - Installation and setup guide
    - Configuration instructions
    - Scheduling setup (cron job or Task Scheduler)
    - Troubleshooting guide
5. Example usage and testing instructions

Out of Scope

- Web interface or dashboard
- Production server deployment

Success Criteria

The delivered script should:

- Successfully merge multiple PDFs into a single file
- Send emails with merged PDF attachment
- Run automatically on schedule
- Handle common error scenarios gracefully
- Be configurable without code changes
- Include clear logs for troubleshooting

Test Plan

Release Name: Updates 01-01
Branch: `updates-01-01`

This document defines the test scenarios, prerequisites, and verification steps for the new features and enhancements deployed in the Updates 01-01 release. The objective is to validate functional correctness, data accuracy, and access control behavior.

Feature 1: Savings Deposit – Branch Field

Objective

Verify that the Branch field is visible and automatically populated on Savings Deposit transactions.

Prerequisites

- A Savings Account linked to a Customer assigned to a specific Branch.

Test Steps and Expected Results

Step	Action	Expected Result
1	Navigate to **Savings > Transactions > Deposits**	Deposit list view opens
2	Click **New**	New deposit form opens
3	Select a **Savings Account**	Account details load
4	Verify **Branch** field	Branch auto-populates based on account
5	Enter Amount and click **Save**	Deposit saved successfully
6	Return to deposit list view	Record appears in list
7	Verify **Branch** column	Correct branch is displayed

Feature 2: Savings Withdrawal – Branch Field

Objective

Verify that the Branch field is visible and automatically populated on Savings Withdrawal transactions.

Prerequisites

- A Savings Account linked to a Customer assigned to a specific Branch.

Test Steps and Expected Results

Step	Action	Expected Result
1	Navigate to **Savings > Transactions > Withdrawals**	Withdrawal list view opens
2	Click **New**	New withdrawal form opens
3	Select a **Savings Account**	Account information loads
4	Verify **Branch** field	Branch auto-populates in Account Information
5	Enter Amount and click **Save**	Withdrawal saved successfully
6	Return to withdrawal list view	Record appears in list
7	Verify **Branch** column	Correct branch is displayed

Feature 3: Contract Search by Phone or Mobile

Objective

Verify that Loan Contracts can be searched using a customer's Phone or Mobile number.

Prerequisites

- At least one Loan Contract linked to a Customer with a Phone or Mobile number.

Test Steps and Expected Results

Step	Action	Expected Result
1	Navigate to **Loan > All Contracts**	Contract list view opens
2	Enter full phone or mobile number in search bar	Search suggestions appear
3	Select **Search Phone or Mobile**	List filters to matching contracts
4	Verify search results	Only related customer contracts are shown
5	Repeat using partial number	"Contains" search works correctly

---

Feature 4: Loan Aging Report – Search by Phone

Objective

Verify that the Loan Aging Report wizard supports phone-based search and auto-fills related fields.

Prerequisites

- A Loan Client with a valid Phone or Mobile number.

Test Scenarios

Scenario A: Phone to Client

Step	Action	Expected Result
1	Open **Loan Aging Report** wizard	Wizard loads
2	Enter phone or mobile number	Input accepted
3	Press Enter or tab out	Client auto-selected

Scenario B: Client to Phone

Step	Action	Expected Result

1	Clear or reopen wizard	Empty form loads
2	Select Client	Client selected
3	Verify Phone field	Phone or mobile auto-fills

Feature 5: Savings Payment Plan Report – Access Rights

Objective

Verify that users with Savings User access rights can generate the Savings Payment Plan Report without access errors.

Prerequisites

- A user account with **Savings / User** role.

Test Steps and Expected Results

Step	Action	Expected Result
1	Log out from Administrator account	Session ends
2	Log in as Savings User	User dashboard loads
3	Navigate to **Savings > Reports > Savings Payment Plan Report**	Report wizard opens
4	Enter required parameters	Parameters accepted
5	Generate report	Report displays successfully
6	Verify system behavior	No access error is shown

Test Completion Criteria

- All features behave as expected
- No functional or access-related errors encountered
- Data consistency verified acros

FullName: Mariam omar dirie
Student_ID: HU0003699

Assignment: Software Quality Assurance

Question 1: Understanding Software (4 Marks)

a. Definition of Software and Difference from Hardware

Software is a collection of data, programs, and procedures that instruct a computer on how to perform specific tasks. Unlike hardware, which is the physical components of a computer (e.g., CPU, memory, keyboard), software is intangible and consists of code written in programming languages.

Example: Microsoft Word is software, while the laptop it runs on is hardware.

b. Main Categories of Software

1. **System Software** – Manages hardware and provides a platform for running applications. Exemple: Windows, Linux.
2. **Application Software** – Designed for end-users to perform specific tasks. Example: Microsoft Excel, Photoshop.
3. **Embedded Software** – Built into devices to control their functions. Example: Software in washing machines or automotive control systems.

c. Software Quality vs Physical Product Quality

Ensuring software quality is harder because software is intangible, complex, and often changes during development. Physical products can be visually inspected or measured, while software defects are logical and may appear only under certain conditions. Moreover, user interactions and environments make software testing unpredictable.

Question 2: Causes of Software Errors (4 Marks)

a. Classification of Software Errors

1) **Human Errors** – Mistakes made by developers or users, such as logic errors or incorrect coding.
2) **Process Errors** – Flaws in the software development process, e.g., poor requirements or inadequate reviews.
3) **Tool Errors** – Issues caused by software tools, compilers, or libraries.
4) **Environmental Errors** – Failures due to external factors like hardware faults, power failures, or network issues.

b. Examples

Error Type	Example 1	Example2
Human	Coder writes wrong formula	Forgot to check user input
Process	Missing requirements	No code review done
Tool	Software tool has a bug	Old library causes error
Environmental	Power outage stops system	Slow internet breaks app

c. Preventive Measures

- Code reviews and peer programming
- Clear requirement documentation
- Automated testing and continuous integration
- Using reliable, updated tools
- System monitoring and backup plans

Question 3: Definition and Objectives of SQA (4 Marks)

a. Definition of SQA

Software Quality Assurance (SQA) is a systematic process that ensures software meets defined quality standards and performs reliably throughout its lifecycle.

b. Objectives of SQA

- Ensure compliance with standards and procedures
- Detect and prevent defects early
- Improve development efficiency
- Maintain customer satisfaction
- Provide continuous improvement feedback loops

c. SQA vs Testing

Testing is one phase within SQA focused on finding bugs in a product. SQA encompasses the entire development process — including planning, process audits, documentation, and post-release evaluations.

Example:

- **Testing:** Checking login functionality works correctly.
- **SQA:** Verifying that the overall development process ensures security, usability, and compliance.

Question 4: Software Quality Factors (4 Marks)

a. Key Quality Factors

1. **Correctness** – Accuracy of output against requirements.
2. **Reliability** – Ability to operate without failure over time.
3. **Usability** – Ease of use and user experience.

4. **Efficiency** – Optimal resource utilization and performance.
5. **Maintainability** – Ease of modifying and fixing software.
6. **Portability** – Ability to function across environments.

b. Example Assessment: E-Commerce Website (e.g., Amazon)

- ***Correctness****:* Displays accurate prices and order details.
- ***Reliability****:* Rarely experiences downtime.
- ***Usability****:* User-friendly design with clear navigation.
- ***Efficiency****:* Fast page loading and optimized search.
- ***Maintainability****:* Regular updates and bug fixes.
- ***Portability****:* Works across browsers and mobile devices.

c. Most Critical Factor in Modern Systems

Reliability is the most critical, as users expect systems — particularly financial, healthcare, or cloud services — to operate continuously without failures that could cause data loss or service disruption.

Question 5: Factors Affecting Intensity of Quality Assurance (4 Marks)

a. Key Factors Influencing SQA Intensity

1. **Project Size** – Larger projects need more testing and documentation.
2. **Complexity** – Complex algorithms or integrations increase QA effort.
3. **Tools and Technology** – Use of automated tools can reduce manual workload.
4. **Team Skills** – Skilled developers and testers produce higher-quality outputs.
5. **Risk Level** – High-risk systems demand more rigorous assurance.

b. Comparison: Critical vs General Applications

- *Critical Systems (e.g., Healthcare, Banking):* Require formal verification, redundancy, and compliance with standards (e.g., ISO 9001).
- *General Applications (e.g., Mobile Games):* Focus more on usability and quick iteration rather than formal testing.

c. Balancing Cost and Quality

- Implement risk-based testing — allocate effort based on impact.
- Automate repetitive tests.
- Adopt Agile and DevOps practices for continuous integration.
- Maintain minimum viable documentation for lean efficiency.

Question 6: Case Study – Ariane 5 Rocket Failure (5 Marks)

a. Category of Error

The failure was primarily a **process error** — the reuse of Ariane 4's software without adequate adaptation or testing for Ariane 5's different flight dynamics. It also included a **human error** in overlooking this incompatibility.

b. Role of SQA in Prevention

Proper SQA reviews, simulation testing, and validation against new requirements could have revealed the integer overflow. Independent verification and validation (IV&V) would have detected the mismatch during pre-launch analysis.

c. Quality Factors Compromised

- **Reliability:** System failed catastrophically under real conditions.
- **Maintainability:** Lack of adaptable design prevented safe reuse.

d. Lessons Learned

- Conduct system-specific testing even for reused software.
- Implement strong SQA review gates before deployment.
- Prioritize fail-safe mechanisms and defensive programming.
- Treat software as a critical component of safety systems, not a secondary concern.

🔗 Suggested Links / Resources

Reference's	Possible Links or Ressource
Pressman, R. S., & Maxim, B. R. (2020). *Software engineering: A practitioner's approach* (9th ed.). McGowan-Hill.	The publisher page for this edition: *Software Engineering: A Practitioner's Approach* on McGowan-Hill site McGraw Hill+1
European Space Agence (ESA). (1996). *Ariane 5 Flight 501 failure report*	ESA press release / inquiry board summary: *Ariane 501 – Presentation of Inquiry Board report* European Space Agency Also, full versions of the report are mirrored (e.g. via MIT / NASA forums) sunnyday.mit.edu+1
Gérard Le Lann. (1996). *The Ariane 5 Flight 501 Failure — A Case Study in System Engineering for Computing Systems*	INRIA / research report versions available online (via ResearchGate, INRIA archives) ResearchGate+2ResearchGate+2

📄 Workflow Report: "Content Idea Generator"

This report details the structure, purpose, and function of the automated workflow named **"Content Idea Generator"**. This workflow is a sophisticated content pipeline designed to harvest news from an automation-focused RSS feed, transform the articles into structured YouTube video concepts using a large language model (LLM), and record the results for production planning.

---

1. Workflow Summary

The **"Content Idea Generator"** workflow is an end-to-end automation process comprising ten connected nodes. Its primary objective is to streamline the content brainstorming process for a YouTube channel specializing in AI and automation.

- **Input Source:** The RSS feed for the `/r/Automation` subreddit.
- **Core Action:** An **AI Agent** powered by a Groq Chat Model generates three structured video ideas per article.
- **Output Destination:** A Google Sheet named **"youtube idea generator"** for tracking and production.

2. Node-by-Node Analysis

The workflow executes in a linear fashion, with distinct phases for data ingestion, cleaning, processing, and output.

Phase 1: Data Ingestion and Normalization (Nodes 1-4)

Node Name	Node Type	Description
When clicking ‘Execute workflow’	`manualTrigger`	Serves as the starting point, manually initiating the process.
HTTP Request	`httpRequest`	Fetches the raw XML data (RSS feed) from `https://www.reddit.com/r/Automation/.rss`.

XML	xml	Converts the fetched XML content into a structured, machine-readable JSON format.
Code	code	Executes a robust custom script to parse the varied structures of RSS/Atom JSON output, extracting and standardizing the **title**, **link**, and **pubDate** for all article entries.

Phase 2: AI Processing and Content Generation (Nodes 5-9)

Node Name	Node Type	Description
Code1	code	Filters the data to a minimalist payload, passing only the **title** and **link** for each article to the AI Agent, reducing token usage and focusing the input.
AI Agent	agent	The central component. It uses the input article details and a detailed **System Message** (acting as an experienced YouTube content strategist) to generate **three unique video concepts** per item.
Groq Chat Model	lmChatGroq	The powerful language model (openai/gpt-oss-20b) that executes the creative generation logic defined by the AI Agent's prompt.
Simple Memory	memoryBufferWindow	Provides short-term memory (30 messages) for the AI Agent, ensuring context is maintained during potentially complex or multi-step interactions.

Code2	code	A critical cleaning step that **parses the raw text output** from the AI Agent. It specifically removes JSON delimiters (\``json`) and converts the resulting string into a structured array of JSON objects.

Phase 3: Final Output (Node 10)

Node Name	Node Type	Description
Google Sheets	googleSheets	Takes the cleaned and structured JSON output and performs an **append** operation to save the video ideas—specifically **videoTitle**, **description**, **hook**, and **format**—into the "**Sheet1**" of the "**youtube idea generator**" spreadsheet.

3. Conclusion

The **"Content Idea Generator"** workflow successfully automates a complex content strategy process. It reliably converts disparate web data into a clean, actionable dataset ready for immediate use by a content team, demonstrating effective use of data transformation and advanced language model capabilities within a unified automation platform.

✅ Top 3 APIs for n8n + Scraping/Search

1. Apify
    - Integration with n8n: n8n has a built-in “Apify” node. You can run Apify Actors (scrapers), fetch datasets, or scrape a single URL.
    - What it’s good at:
        - Full browser-based scraping (Playwright / Puppeteer) or lightweight HTTP client → handle complex, JS-heavy e-commerce sites.
        - Use “Actors” (pre-built scrapers) or build custom ones to extract product details, reviews, listings, etc.
        - Automate workflows: schedule scrapes, chain scrapers, trigger other n8n nodes when data is ready.
    - Why choose it: Very flexible, scalable, and well-integrated into n8n for more advanced / large-scale scraping tasks.
2. ScraperAPI
    - Integration with n8n: There is an official / community node for ScraperAPI in n8n.
    - What it’s good at:
        - Simple “send URL, get back HTML (or parsed response)” model.
        - Handles proxy rotation, CAPTCHAs, rendering of JS automatically.
        - You can configure device type (desktop/mobile), geo-targeting, and level of proxy / rendering.
        - Supports integration with n8n’s MCP client for real-time AI agent use.
    - Why choose it: Lightweight and reliable for “on-demand scraping” from workflows. If your agent just needs to fetch product pages and extract info, this is easy to plug in.
3. ScrapingBee

- Integration with n8n: There is a verified ScrapingBee integration in n8n.
- What it's good at:
  - Web scraping API that supports JavaScript rendering, rotating proxies, and CAPTCHAs.
  - Good for extracting structured data (e.g., product details from Amazon, Alibaba) because you can reliably fetch fully rendered HTML.
  - Their Amazon-specific endpoint helps with product APIs (ScrapingBee has Amazon Search / Amazon Product API).
- Why choose it: Balanced between power and simplicity: more robust than plain HTTP + proxies, but lighter than full-blown browser automation.

⚠️ Bonus Option to Consider

- Scrapfly: They have an official n8n integration. Their Web Scraping API supports browser rendering, anti-bot, and they also have an Extraction API for parsing product data directly.
- Bright Data: Also supported in n8n. You can use their HTTP API or MCP integration to do large-scale scraping.
- ScrapeNinja: Has an n8n community node for browser + JS scraping, extracting content via custom JS.

💡 My Recommendation (for AI Agents that Search + Scrape)

- Use Apify for your core scraping: run actors to get product pages, crawl, extract structured product data — fits well in an "agent picks a URL → scrape → return JSON → agent reasons."
- Use ScraperAPI for lighter "on-the-fly" scraping from agents when they need a quick look at a page or search result.

- Use ScrapingBee when you specifically want product data from e-commerce sites and need JS rendering + stable proxy but don’t need full actor control.

📄 Workflow Report: "Content Idea Generator"

This report details the structure, purpose, and function of the automated workflow named **"Content Idea Generator"**. This workflow is a sophisticated content pipeline designed to harvest news from an automation-focused RSS feed, transform the articles into structured YouTube video concepts using a large language model (LLM), and record the results for production planning.

---

1. Workflow Summary

The **"Content Idea Generator"** workflow is an end-to-end automation process comprising ten connected nodes. Its primary objective is to streamline the content brainstorming process for a YouTube channel specializing in AI and automation.

- **Input Source:** The RSS feed for the `/r/Automation` subreddit.
- **Core Action:** An **AI Agent** powered by a Groq Chat Model generates three structured video ideas per article.
- **Output Destination:** A Google Sheet named **"youtube idea generator"** for tracking and production.

2. Node-by-Node Analysis

The workflow executes in a linear fashion, with distinct phases for data ingestion, cleaning, processing, and output.

Phase 1: Data Ingestion and Normalization (Nodes 1-4)

Node Name	Node Type	Description
When clicking ‘Execute workflow’	`manualTrigger`	Serves as the starting point, manually initiating the process.
HTTP Request	`httpRequest`	Fetches the raw XML data (RSS feed) from `https://www.reddit.com/r/Automation/.rss`.

XML	xml	Converts the fetched XML content into a structured, machine-readable JSON format.
Code	code	Executes a robust custom script to parse the varied structures of RSS/Atom JSON output, extracting and standardizing the **title**, **link**, and **pubDate** for all article entries.

Phase 2: AI Processing and Content Generation (Nodes 5-9)

Node Name	Node Type	Description
Code1	code	Filters the data to a minimalist payload, passing only the **title** and **link** for each article to the AI Agent, reducing token usage and focusing the input.
AI Agent	agent	The central component. It uses the input article details and a detailed **System Message** (acting as an experienced YouTube content strategist) to generate **three unique video concepts** per item.
Groq Chat Model	lmChatGroq	The powerful language model (openai/gpt-oss-20b) that executes the creative generation logic defined by the AI Agent's prompt.
Simple Memory	memoryBufferWindow	Provides short-term memory (30 messages) for the AI Agent, ensuring context is maintained during potentially complex or multi-step interactions.

Code2	code	A critical cleaning step that **parses the raw text output** from the AI Agent. It specifically removes JSON delimiters (\``json`) and converts the resulting string into a structured array of JSON objects.

Phase 3: Final Output (Node 10)

Node Name	Node Type	Description
Google Sheets	googleSheets	Takes the cleaned and structured JSON output and performs an **append** operation to save the video ideas—specifically **videoTitle**, **description**, **hook**, and **format**—into the "**Sheet1**" of the "**youtube idea generator**" spreadsheet.

3. Conclusion

The **"Content Idea Generator"** workflow successfully automates a complex content strategy process. It reliably converts disparate web data into a clean, actionable dataset ready for immediate use by a content team, demonstrating effective use of data transformation and advanced language model capabilities within a unified automation platform.