

How an AI Agent Built This: A Technical Replication Guide

Author: Matrix Agent
Date: 28 January 2026
Document Type: Technical Implementation & Replication Guide

Table of Contents

- 1. [What Is Matrix Agent?](#)
- 2. [Capabilities & Tool Access](#)
- 3. [How I Approached This Project](#)
- 4. [Phase-by-Phase Technical Breakdown](#)
- 5. [Code & Commands Executed](#)
- 6. [Decision-Making Process](#)
- 7. [Error Handling & Problem Solving](#)
- 8. [Human Replication Guide](#)
- 9. [Tools & Environment Setup](#)
- 10. [Lessons Learned](#)

1. What Is Matrix Agent?

1.1 Identity

I am **Matrix Agent**, an AI assistant developed to handle complex, multi-step tasks autonomously. I operate within a conversational interface but have access to real tools that execute on the user's local machine.

1.2 How I Differ From Standard Chatbots

Standard Chatbot	Matrix Agent
Generates text responses only	Executes real commands on your system
Cannot access files	Reads, writes, and edits files directly
Cannot run code	Executes PowerShell/Bash commands

Standard Chatbot	Matrix Agent
No persistence	Creates files that persist after conversation
No external integrations	Connects to GitHub, web APIs, and more

1.3 My Operating Environment

For this project, I operated on:

- **OS:** Windows (detected via PowerShell responses)
- **User Directory:** `C:\Users\Nana_Editing\`
- **Working Directory:** `C:\Users\Nana_Editing\Downloads\`
- **Shell:** PowerShell (not CMD)

2. Capabilities & Tool Access

2.1 File System Tools

Tool	What It Does	How I Used It
<code>Read</code>	Read file contents	Examined XML files, chat files
<code>Write</code>	Create/overwrite files	Created .tex, .json, .md files
<code>Edit</code>	Find and replace in files	Modified specific content
<code>Glob</code>	Pattern-match file search	Found all <code>*.md</code> files
<code>Grep</code>	Search file contents	Located WhatsApp chat patterns

2.2 Command Execution

Tool	What It Does	How I Used It
<code>bash</code>	Execute shell commands	Ran git, gh, PowerShell commands
<code>bash_output</code>	Get output from background processes	Monitored long-running tasks

2.3 Conversion & Generation

Tool	What It Does	How I Used It
<code>convert_file</code>	Convert between formats	Markdown → PDF
<code>convert_docx_to_md</code>	DOCX to Markdown	(Available but used manual XML extraction)

2.4 External Integrations

Tool	What It Does	How I Used It
<code>batch_web_search</code>	Search the web	(Available for research)
<code>extract_content_from_websites</code>	Scrape web pages	(Available for research)

3. How I Approached This Project

3.1 My Reasoning Process

When the user made a request, I followed this mental model:

1. UNDERSTAND → What exactly does the user want?
2. PLAN → What steps are needed?
3. DISCOVER → What exists already? (scan files, check tools)
4. EXECUTE → Run commands, create files
5. VERIFY → Did it work? Check outputs
6. ADAPT → If errors, try alternative approaches
7. REPORT → Summarise what was done

3.2 Context Continuity

This project spanned multiple conversation turns. I maintained context by:

- Remembering file paths created earlier
- Tracking which tasks were completed
- Building on previous outputs (e.g., using extracted XML to create LaTeX)

3.3 User Intent Interpretation

The user's requests were often conversational. I translated them into technical tasks:

User Said	I Interpreted As
"crawl through all the .md files"	Use <code>Glob</code> to find <code>**/*.md</code> , then <code>Read</code> each
"prep the files for GitHub"	Rename files to naming convention, copy to folder
"submit them to GitHub for me"	Use <code>gh</code> CLI to authenticate, commit, push
"can u do any of this for me?"	Parse the workflow document and execute it
"don't touch the content"	Preserve exact text, only change format

4. Phase-by-Phase Technical Breakdown

4.1 Phase 1: WhatsApp Chat Discovery

Goal: Find all WhatsApp chat files converted to `.md`

Method:

```
Tool: Glob
Pattern: **/*.md
Path: C:\Users\Nana_Editing\
```

Result: Found 44 `.md` files, filtered to 21 WhatsApp chats by examining content patterns.

How I Identified WhatsApp Chats:

- Looked for pattern: `[DD/MM/YYYY, HH:MM:SS] Name: Message`
- Checked filenames containing "WhatsApp Chat"

4.2 Phase 2: Documentation Creation

Goal: Create a folder explaining the project

Method:

```
Tool: bash (PowerShell)
Command: New-Item -ItemType Directory -Path "...
\WhatsApp_Chat_Analysis"

Tool: Write
File: WhatsApp_Chats_Overview.md
Content: [Structured documentation]
```

Programmatic Approach:

I generated markdown content programmatically by:

1. Listing all discovered files
 2. Categorising by organisation (extracted from filenames)
 3. Describing the format with examples
 4. Writing to a new file
-

4.3 Phase 3: File Renaming & Preparation

Goal: Rename files to `YYYY-MM-DD_[project]_[chat-type].txt` convention

Method:

Tool: bash (PowerShell)

Commands:

```
New-Item -ItemType Directory -Force -Path "...\\unprocessed"
Copy-Item "source.md" -Destination "...\\unprocessed\\renamed.txt"
```

Why Copy Instead of Rename:

- Preserves originals (non-destructive)
 - Allows rollback if naming was wrong
 - User explicitly wanted originals untouched
-

4.4 Phase 4: GitHub Integration

Goal: Push files to user's GitHub repository

Step 4.4.1: Install Git

```
winget install --id Git.Git -e --source winget
```

Step 4.4.2: Authenticate GitHub CLI

```
gh auth login
# Selected: GitHub.com → HTTPS → Web browser
# User entered one-time code: 871B-FD6A
```

Step 4.4.3: Clone, Add, Commit, Push

```
git clone https://github.com/andiekobbietks/Curriculum-Vitae-life-experiences-repo.git
cd repo
# Copy files to appropriate directories
git add .
git config user.email "andiekobbietks@outlook.com"
git config user.name "Andrea Enning"
git commit -m "Add files"
git push origin main
```

4.5 Phase 5: CV Conversion Pipeline

Goal: Convert DOCX to LaTeX preserving brand identity

Step 4.5.1: Extract XML from DOCX

DOCX files are ZIP archives. I extracted them:

```
Copy-Item "CV.docx" "cv.zip"
Expand-Archive -Path "cv.zip" -DestinationPath "docx_xml"
```

Step 4.5.2: Analyse XML Structure

```
Tool: Read
Files examined:
- word/document.xml (main content)
- word/styles.xml (formatting)
- word/theme/theme1.xml (colours)
- word/fontTable.xml (fonts)
```

Step 4.5.3: Extract Design Tokens

I parsed XML to find:

- Colours: `<a:srgbClr val="025940"/>` → `#025940`
- Fonts: `<w:rFonts w:ascii="Arial"/>` → Arial

Step 4.5.4: Generate LaTeX

```
Tool: Write
File: andrea_enning_cv.tex
Content: [Complete LaTeX document with extracted tokens]
```

Key LaTeX Setup:

```
\definecolor{primary}{HTML}{025940}  
\setmainfont{Arial}
```

4.6 Phase 6: GitHub Copilot Agent Task

Goal: Create automated task to enrich CV from WhatsApp chats

Method:

```
gh extension install github/gh-copilot  
gh agent-task create "Read WhatsApp chats... enrich CV..."
```

Output:

- Created PR #5
 - Agent session started
 - URL provided for monitoring
-

5. Code & Commands Executed

5.1 Complete Command Log

Below is every significant command I executed:

```

# Phase 1: Discovery
# (Used internal Glob tool, not shell command)

# Phase 3: File Preparation
New-Item -ItemType Directory -Force -Path "C:\Users\Nana_Editing\Downloads\WhatsApp_Chat_Analysis\unprocessed"
Copy-Item "WhatsApp Chat - Technocamps.md" -Destination "...2024-XX-XX_technocamps_group.txt"
# (Repeated for 18 files)

# Phase 4: GitHub Setup
winget install --id Git.Git -e --source winget
gh auth login
git clone https://github.com/andiekobbietks/Curriculum-Vitae-life-experiences-repo.git

# Phase 4: Git Operations
cd repo
git add .
git config user.email "andiekobbietks@outlook.com"
git config user.name "Andrea Enning"
git commit -m "Add WhatsApp chat files"
git push origin main

# Phase 5: DOCX Extraction
Copy-Item "Andrea Enning Updated CV.docx" "cv.zip"
Expand-Archive -Path "cv.zip" -DestinationPath "docx_xml"

# Phase 6: Copilot Agent
gh extension install github/gh-copilot
gh agent-task create "..."

```

5.2 Files I Created Programmatically

File	Method	Size
WhatsApp_Chats_Overview.md	Write tool	~2 KB
Rename_Guide_for_Processing.md	Write tool	~3 KB
cv_tokens.json	Write tool	894 B

File	Method	Size
<code>cv_content.json</code>	Write tool	7.7 KB
<code>cv_template.tex</code>	Write tool	3.3 KB
<code>andrea_enning_cv.tex</code>	Write tool	9.4 KB
<code>Project_Report.md</code>	Write tool	15.4 KB
<code>Project_Report.pdf</code>	convert_file tool	~200 KB

6. Decision-Making Process

6.1 Why I Made Certain Choices

Decision	Reasoning
Used PowerShell, not CMD	Windows environment detected; PowerShell is more capable
Copied files instead of moving	Non-destructive; preserves originals
Extracted XML manually	More control than automated DOCX converters
Created JSON tokens file	Reusable for future templates
Used <code>gh agent-task</code> not web UI	User wanted CLI automation

6.2 When I Asked for Clarification

I asked the user when:

- Multiple valid approaches existed
- The request was ambiguous
- Destructive operations were involved

Example: When the user said "just the originals," I confirmed they meant 18 files without duplicates before pushing.

6.3 When I Made Autonomous Decisions

I proceeded without asking when:

- The path was clear from context
- The operation was reversible
- Industry best practices applied

Example: I chose `processed-outputs/latex-cv/` as the folder structure without asking, as it follows logical organisation.

7. Error Handling & Problem Solving

7.1 Errors Encountered & Solutions

Error	Cause	Solution
<code>'&&' is not a valid statement separator</code>	PowerShell syntax differs from bash	Changed <code>&&</code> to <code>;</code>
<code>Expand-Archive</code> failed on <code>.docx</code>	Extension not recognised as ZIP	Copied file to <code>.zip</code> first
<code>Author identity unknown</code>	Git not configured	Ran <code>git config user.email/name</code>
<code>unknown command "copilot"</code>	Extension not installed	Ran <code>gh extension install github/gh-copilot</code>
Character encoding in filename (André)	Unicode path issues	Used wildcard <code>Andr*</code>

7.2 My Error Recovery Process

1. Detect error from command output
2. Analyse error message
3. Research solution (from training knowledge)
4. Attempt fix
5. If fix fails, try alternative approach
6. If still failing, explain to user and ask for guidance

8. Human Replication Guide

8.1 Prerequisites

To replicate this workflow manually, you need:

Tool	Installation
Git	<code>winget install --id Git.Git</code> (Windows) or <code>brew install git</code> (Mac)
GitHub CLI	<code>winget install --id GitHub.cli</code> or <code>brew install gh</code>
Text Editor	VS Code, Notepad++, or similar
PowerShell/ Terminal	Built into OS

8.2 Step-by-Step Manual Replication

Step 1: Organise WhatsApp Exports

```
# Create folder structure
mkdir WhatsApp_Chat_Analysis
mkdir WhatsApp_Chat_Analysis\unprocessed

# Copy and rename your chat exports
copy "WhatsApp Chat - Project.txt"
"WhatsApp_Chat_Analysis\unprocessed\2024-01-15_project_group.txt"
```

Step 2: Set Up GitHub Repository

```
# Authenticate
gh auth login

# Create repository (or clone existing)
gh repo create my-cv-repo --private
cd my-cv-repo

# Create folder structure
mkdir raw-inputs\whatsapp-chats\unprocessed
mkdir processed-outputs\latex-cv
```

Step 3: Push WhatsApp Chats

```
# Copy files to repo
copy "...\\WhatsApp_Chat_Analysis\\unprocessed\\" "raw-inputs\\whatsapp-
chats\\unprocessed\\"

# Commit and push
git add .
git commit -m "Add WhatsApp chat exports"
git push origin main
```

Step 4: Convert CV to LaTeX

Manual Approach:

1. Rename your `.docx` to `.zip`
2. Extract the ZIP
3. Open `word/document.xml` in a text editor
4. Copy text content to a LaTeX template
5. Open `word/theme/theme1.xml` to find brand colours
6. Update LaTeX `\definecolor` commands

Or use an online converter like:

- docx2latex.com
- pandoc: `pandoc cv.docx -o cv.tex`

Step 5: Create Copilot Agent Task

```
# Install Copilot extension
gh extension install github/gh-copilot

# Navigate to your repo
cd my-cv-repo

# Create the task
gh agent-task create "Read files in raw-inputs/whatsapp-chats/ and use
the information to enrich my CV in processed-outputs/latex-cv/cv.tex"
```

Step 6: Review and Merge

1. Open the PR link provided by the agent
2. Review proposed changes
3. Request modifications if needed

9. Tools & Environment Setup

9.1 Windows Setup Script

```
# Install required tools
winget install --id Git.Git -e --source winget
winget install --id GitHub.cli -e --source winget
winget install --id Microsoft.VisualStudioCode -e --source winget

# Configure Git
git config --global user.name "Your Name"
git config --global user.email "your@email.com"

# Authenticate GitHub
gh auth login

# Install Copilot extension
gh extension install github/gh-copilot
```

9.2 Mac/Linux Setup Script

```
# Install Homebrew (if not installed)
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install required tools
brew install git
brew install gh

# Configure Git
git config --global user.name "Your Name"
git config --global user.email "your@email.com"

# Authenticate GitHub
gh auth login

# Install Copilot extension
gh extension install github/gh-copilot
```

9.3 Recommended VS Code Extensions

- LaTeX Workshop (for `.tex` editing)
 - GitHub Copilot (for AI assistance)
 - Markdown Preview Enhanced (for `.md` files)
-

10. Lessons Learned

10.1 What Worked Well

Aspect	Why It Worked
Iterative approach	Building step-by-step allowed error correction
Non-destructive operations	Copying instead of moving prevented data loss
Version control first	Git tracked all changes for easy rollback
Structured prompts	Clear agent task instructions produced better results

10.2 What Could Be Improved

Aspect	Improvement
DOCX parsing	Could use dedicated library (python-docx) for cleaner extraction
Chat preprocessing	Could clean/anonymise chats before pushing
Template system	Could create reusable LaTeX templates
Automation	Could script entire workflow for one-click execution

10.3 Key Takeaways for Humans

1. **AI agents are tools, not magic** - I execute commands you could run yourself
2. **Understand what's happening** - Don't blindly trust; review outputs
3. **Version control everything** - Git is your safety net
4. **Iterate and verify** - Check results at each step
5. **Keep originals** - Never modify source files directly

Conclusion

This document has revealed the "how" behind the project—showing that an AI agent is fundamentally a sophisticated command executor with natural language understanding. Everything I did can be replicated by a human with the right tools and knowledge.

The value I provide is:

- **Speed** - I execute faster than manual typing
- **Knowledge** - I know command syntax and best practices
- **Persistence** - I don't forget context mid-task
- **Adaptation** - I adjust when errors occur

But ultimately, I am running the same tools available to any developer. This guide empowers you to replicate, modify, and extend this workflow independently.

Technical guide generated by Matrix Agent on 28 January 2026