Election Watch by CCIJ

Step-by-Step Playbook

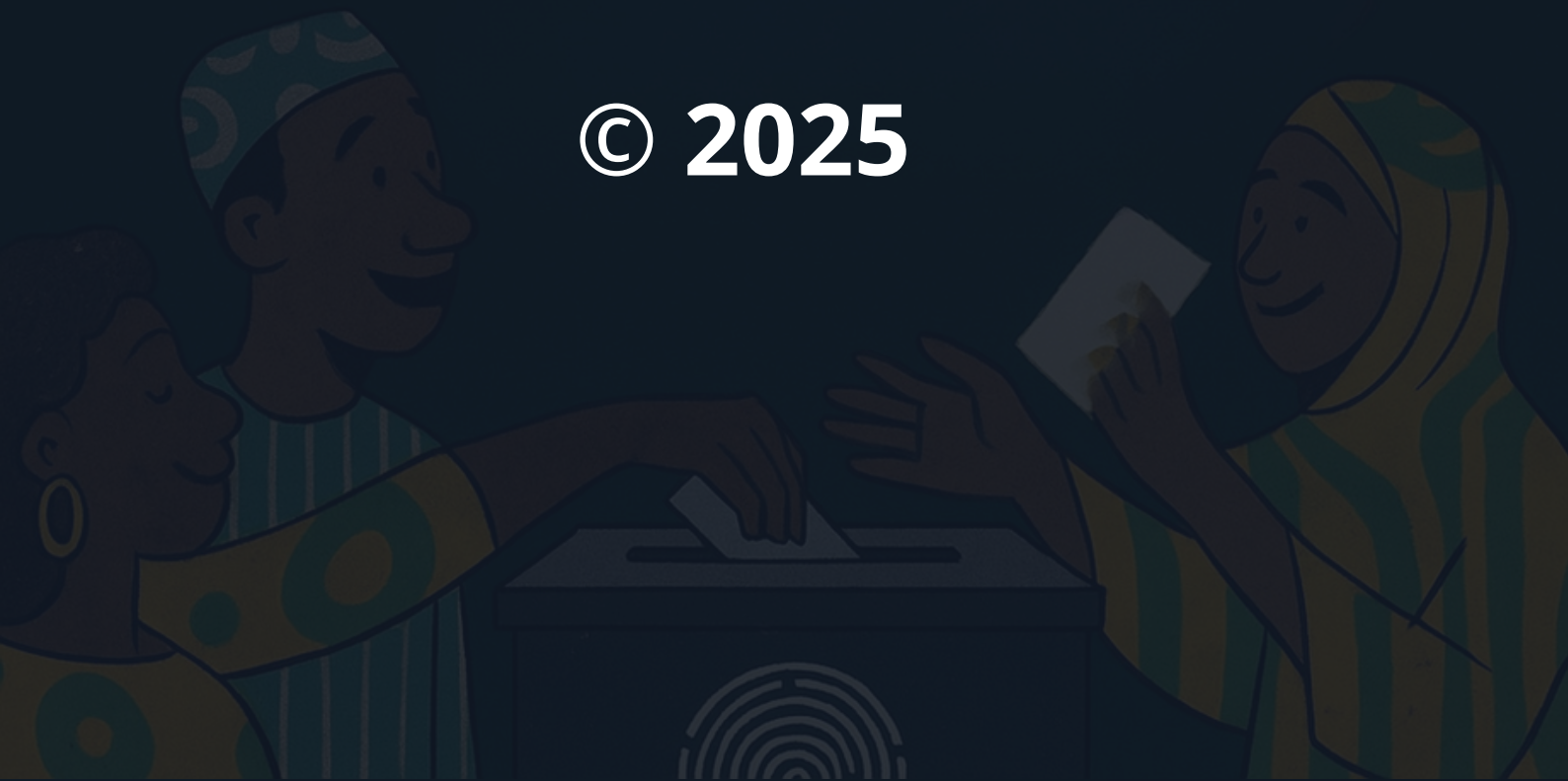


Table Of Contents

Opening Credits & Acknowledgements

The ElectionWatch ML Multi-Agent System was made possible through the collaboration, expertise, and dedication of an interdisciplinary team committed to safeguarding electoral integrity across Africa

Project Leadership

- **Commissioning Partner:** Jeff Kelly Lowenstein (CCIJ)
- **Product Manager:** Nelly Kalu (CCIJ)
- **Project Manager (PM)**: Nelly Kalu (CCIJ), Mwende Mukwanyaga, Lilian Mutinda (AI Salon Kenya) -Oversaw timelines, deliverables, and team coordination
- **Technical Lead:** Robin Kiplangat-Led system architecture, API design, and ML integration
- **Data Leads:** J.M Okari, Sotiris Sideris- Led data collection, cleaning and validation
- **Ethics & Governance Lead:** Mwende Mukwanyaga -Ensured adherence to fairness, transparency, and privacy principles
- **Quality Assurance Lead:** Hesbon Ombati - Directed testing, bug tracking, and release readiness
- **Stakeholder Engagement:** Fola Folayan, Lilian Mutinda- Liaises with journalists, election observers, civic groups.

Core Development Team

- **Backend Engineer:** Emery Bashige- Built FastAPI services, database logic, and API endpoints
- **Frontend Engineer:** Vincent Kipyegon- Created user interface, dashboards, and interactive visualisation
- **Machine Learning Engineers:** Robin Kiplangat- Developed narrative extraction, NER, and relationship models
- **Data Engineers:** Mary Wairimu, Lilian Mutinda, Alice Giorgio, Henrik Vatndal-Designed and implemented data pipelines and pre-processing systems
- **DevOps Engineers:** Robin Kiplangat, Victor Mwendwa- Managed deployment, CI/CD pipelines, and infrastructure
- **UX/UI Designers:** Charles Assam- Crafted the user experience and design system
- **Handover & Transition Manager:** Jillian Dudziak- overseeing the migration of systems, knowledge, and documentation from the project team to the client/end-users.

Core Development Team

- **Backend Engineer:** Emery Bashige- Built FastAPI services, database logic, and API endpoints
- **Frontend Engineer:** Vincent Kipyegon- Created user interface, dashboards, and interactive visualisation
- **Machine Learning Engineers:** Robin Kiplangat- Developed narrative extraction, NER, and relationship models
- **Data Engineers**: Mary Wairimu, Lilian Mutinda, Alice Giorgio, Henrik Vatndal-Designed and implemented data pipelines and pre-processing systems
- **DevOps Engineers:** Robin Kiplangat, Victor Mwendwa- Managed deployment, CI/CD pipelines, and infrastructure
- **UX/UI Designers:** Charles Assam- Crafted the user experience and design system
- **Handover & Transition Manager:** Jillian Dudziak- overseeing the migration of systems, knowledge, and documentation from the project team to the client/end-users.

Research & Ethics Advisory

- **Translators/local context:** Chukwunaeme Kingsley Obiejesi and Faith Maryjane Oshoko(Pidgin), Promise Ugonna (Igbo), Haruna Ibrahim Kakangi (Hausa), and Faoziyah Saanu (Yoruba.)
- **Fairness & Bias Advisors:** Mwende Mukwanyaga, Hesbon Ombati - Reviewed ML outputs for bias, drift, and equity risks
- **Data Privacy Advisor:** Doreen Mwikali - Oversaw compliance with data protection policies
- **Legal & Governance Advisor:** Advocate Stephanie Marigu - Provided guidance on legal frameworks for elections data

Contributors & Reviewers

- **Documentation Writers:** Mwende Mukwanyaga, Sotiris Sideris, Alice Giorgio, J.M Omari, Mary Wairimu, Robin Kiplangat, Vincent Kipyegon - Created the playbook, API guides, and internal developer docs
- **Documentation Editors/Reviewers:** Lilian Mutinda, Nelly Kalu
- **QA Officer:** Hesbon Ombati- Ran full system tests and verified launch readiness
- **Community Reviewers:** Gaute Kokkvoll (Factiverse), Henrik Vatndal (Factiverse), Fola Folayan, Nigel Mugamu, Ajibola Amzat, Jeff Kelly Lowenstein, Karwitha Kirimi, Alfred Oduor, Shantelle Wanjiru- Offered critical feedback on usability and ethics

We extend our sincere gratitude to JournalismAI for their generous support, without which this project would not have been possible. Their commitment to advancing responsible and innovative uses of Artificial Intelligence in journalism provided both the resources and mentorship needed to bring this vision to life. Our appreciation goes to our partners, testers, and early users, whose active engagement shaped the project at every stage. From providing candid feedback during early trials to validating the outputs against real investigative workflows, your contributions ensured that the system was not only technically sound but also editorially meaningful.

Finally, we recognise the dedication of the technical and editorial teams who invested their expertise, time, and creativity in designing, building, and refining the platform. This collaboration between technologists and journalists exemplifies the kind of cross-sector partnership needed to strengthen the future of independent media.

Chapter 01 Project Overview

The purpose of ElectionWatch is simple: Create a network analysis of electoral disinformation; Identify the key actors; Show the timeline of spread. The aim is to strengthen democratic resilience by analyzing and documenting disinformation and misinformation in electoral contexts.
It supports and improves election reporting and monitoring by leveraging data-driven insights to counter false information, enhance transparency, and safeguard press freedom during electoral processes in West Africa and beyond.
Elections in the digital age are often accompanied by a flood of false or misleading information, from coordinated disinformation campaigns to grassroots misinformation spread through community platforms. These narratives can distort public opinion, erode trust in institutions, and in some cases, fuel violence.
Built by journalists to provide journalists, civil society, and other stakeholders with the means to detect, contextualise, and respond to such information risks, Election Watch helps to close the gap between the speed of misinformation spread and the capacity of stakeholders to understand and act on it by combining data engineering, open-source intelligence, and machine learning,
ElectionWatch defines electoral disinformation as the deliberate creation and dissemination of false or misleading information—including factually accurate content taken out of context (malinformation)—with the intent to manipulate voter perceptions, influence election outcomes, undermine trust in electoral processes and institutions, or harm individuals or groups involved in elections.

Election risks addressed include:

- Disinformation campaigns targeting candidates, institutions, or voters
- Viral misinformation narratives that spreads unchecked
- Information vacuums during sensitive periods (vote tallying, disputes, violence)
- Use of coded language, hate speech, or polarising narratives that can incite harm
- Malicious or misleading content deployed in political mudslinging and adversarial attacks

Objectives

- Track and map narratives and counter-narratives to understand how information flows and evolves.
- Generate timelines of events and actors to provide chronological context and evidence trails.
- Produce actionable reports for stakeholders, including journalists, fact-checkers, election monitors, and civil society organisations, so they can make informed interventions.

Success Metrics

Success will be measured both technically (system performance) and operationally (impact on users and investigations).

Technical metrics:

- Volume of data successfully ingested, cleaned, and stored per election cycle
- Accuracy of narrative extraction and entity recognition (precision/recall benchmarks)
- Stability and uptime of system infrastructure during peak periods
- Number of reports generated automatically via the platform

Operational/impact metrics:

- Use of Election Watch outputs in investigative stories, fact-checks, or monitoring reports
- Feedback from newsroom and civil society partners on usability and relevance
- Evidence of narratives flagged early enough to allow preventive responses

Scope

In-scope:

- Identifying and mapping narratives, counter-narratives, and coded language
- Constructing timelines of events, actors, and narratives
- Generating structured reports for stakeholders (journalists, monitors, fact-checkers)
- Election-related datasets, including social media, news, OSINT, and messaging platforms

Out-of-scope:

- Direct removal of content from platforms (this is the role of platforms, not ElectionWatch)
- Partisan campaigning, advocacy, or electoral propaganda
- Direct policy lobbying or legislative reform on misinformation law
- Real-time voter influence operations (e.g., microtargeting voters with content)
- General analysis of non-election related topics (e.g., celebrity gossip, sports rumours

Chapter 02 Team Structure

Below is a practical, copy-pasteable playbook you can use to put up Project Leadership and the cross-functional team. These are placeholders where you can add names later, plus concrete checklists, meeting cadence, KPIs and a tiny RACI example so nothing is ambiguous.

2.1 Create the master team roster

a. Make a single spreadsheet called Team_Roster.xlsx with these columns: Role, Name (or [TBD]), Pronouns, Start date, Contact, Location, Notes.

Role	Name	Pronouns	SD	Contact	Location	Notes

b. Populate the four Project Leadership roles immediately. Example rows:

Project/Product Manager: [Name or TBD]
Technical Lead: [Name or TBD]
Ethics & Governance Lead: [Name or TBD]
Quality Assurance Lead: [Name or TBD]

c. Add a second tab for extended roles (ML Engineer, Data Engineer, Data Analysts, Editors, Backend, Frontend, UX, DevOps, Legal, Product Owner, Technical Writer, Community Manager) and mark priority hires.

Core Functions

- **Data Acquisition & Pre-Processing Team (DA)**: Builds pipelines, cleans and validates data
- **Machine Learning Engineers (MLE)**: Develop narrative extraction, NER, relationship models
- **Backend Developers:** Build APIs, storage, logic for timelines and networks
- **Frontend/UX Designers:** Design interface, graphs, and timeline visualisation
- **Security & DevOps:** Manage infrastructure, CI/CD, monitoring, compliance
- **Analysts / Researchers:** Monitor narratives, annotate datasets, support reporting
- **Editorial & Communications:** Package findings into briefs, alerts, and reports

Appoint role owners or placeholders

- If you know people, put their names. If not, write [TBD| Recruit] and pick an interim backup (could be the PM).
- For each role add 1–2 backups to reduce single-point failure risk. Example: Technical Lead backup = Senior Backend Engineer.

2.2 Onboarding checklist (for each new role)

1. Access: repo, issue tracker, cloud console, Slack, calendar invites, product/project management tools.
2. Read: project-plan.md, architecture.md, data-inventory.csv, ethics-checklist.md.
3. Meet: 30-minute intro with PM, 60-minute technical on-boarding with Technical Lead, 45-minute ethics briefing with Ethics Lead.
4. Deliver: within the first week submit a 1-page note: “First 7 days: priorities, blockers, dependencies”.

2.3 Onboarding checklist (for each new role)

For each role below, we list the core responsibilities, a short 7-step starter checklist, and 3 guiding questions they should ask

2.3.1 Project Manager (PM)

Core: oversee timelines, coordinate deliverables, run meetings, manage risks.

7-step starter checklist

1. Create project-plan.md with milestones, owners and dates.
2. Set up cadence: daily standup, weekly tactical, biweekly planning, monthly steering.
3. Create risk register risks.csv.
4. Create release calendar releases.csv.
5. Set up issue templates in tracker (bug, task, improvement).
6. Run first sprint planning and assign tickets.
7. Produce first weekly status update.

Guiding questions

- What is our MVP for the next 30 days?
- Which dependencies could block delivery?
- Who is the backup for each critical role?

KPIs: on-time milestone delivery, percent of blocked tickets resolved within 48 hours.

2.3.2 Technical Lead

Core: system architecture, API spec, ML integration, code reviews, tech decisions.

7-step starter checklist

1. Draft architecture.md and share for review.
2. Create API spec api-spec.yaml (OpenAPI).
3. Decide infra pattern and create infra repo with IaC.
4. Set CI/CD pipelines and deployment checklist.
5. Define code standards and review process.
6. Create an initial end-to-end demo.
7. Document data flows and compute costs estimate.

Guiding questions

- What are single points of failure in architecture?
- How will models be versioned and rolled back?
- What telemetry do we need for observability?

KPIs: mean time to deploy, mean time to recover, test coverage on core services.

2.3.3 Ethics & Governance Lead

Core: fairness, transparency, privacy, consent, regulatory compliance.

7-step starter checklist

1. Create ethics-checklist.md and data-inventory.csv.
2. Run a data protection impact assessment (DPIA) and file it.
3. Produce a model card model-card.md for each ML model.
4. Define logging and audit requirements.
5. Approve data collection consent language.
6. Set up periodic audits and sign-offs for releases.
7. Train the team on minimum privacy practices.

Guiding questions

- Which dataset elements are sensitive and how are they protected?
- What fairness tests are required before release?
- What are legal reporting or consent obligations?

KPIs: number of unresolved ethics findings, time to close DPIA issues.

2.3.4 Quality Assurance Lead

Core: testing strategy, bug triage, release readiness.

7-step starter checklist

1. Draft test-plan.md and acceptance criteria per ticket.
2. Set up a test automation framework and CI test suites.
3. Create exploratory test charters for manual QA.
4. Define severity levels and SLAs for bug fixes.
5. Run a release readiness checklist before any deployment.
6. Create a regression test schedule.
7. Maintain metrics dashboard for pass rates and flakiness.

Guiding questions

- What are the critical user journeys that must not fail?
- Which tests must be green before release?
- How will we handle hotfixes post-release?

KPIs: test pass rate, bug escape rate, mean time to close critical bugs.

2.3.5 Cross-functional role suggestions (hire or assign)

Short list to fill in after leadership: ML Engineer, Data Engineer, Backend Engineer, Frontend Engineer, DevOps, Product Owner, UX Researcher, Legal Counsel, Security Officer, Technical Writer, Translators, Community Manager. Add each to Team_Roster.xlsx with priority level.

2.3.6 RACI for core activities (tiny example)

Define Responsible (R), Accountable (A), Consulted (C), Informed (I) for five activities.

Activity	Responsible (R)	Accountable (A)	Consulted (C)	Informed (I)
Architecture design	Tech Lead	PM	ML Eng, DevOps	Steering Committee
Build & implement	Engineers	Tech Lead	PM, QA	Stakeholders
QA & test	QA Lead	PM	Tech Lead	All
Ethics review	Ethics Lead	PM	Legal, Tech Lead	All
Release to prod	DevOps	PM	Tech Lead, QA	Stakeholders

(Use this as a template and expand in your project tracker.)

2.4 Meeting cadence and artifacts

2.4.1 Communication & Reporting

Channels: (Whatsapp/Discord/Signal/Email)

Meeting Rhythm: (daily standup, weekly sprint review, monthly steering)

Documentation: (Fireflies notetaker/Notion/GitHub wiki, Basecamp)

Reporting Cadence: (weekly internal progress update, monthly stakeholder report)

a. Daily standup : 15 minutes to focus on blockers.
Artifact: standup-notes.md.

b .Weekly tactical: 60 minutes of sprint progress, risks, dependencies.
Artifact: weekly-status.md.

c. Biweekly sprint planning: 90 minutes to scope the next sprint, define acceptance.
Artifact: sprint board/trello cards

d. Monthly steering: 60 minutes with exec stakeholders, budget, strategy.
Artifact: steering deck.

e. Post-release retrospective: 60 minutes and have action items to prevent regression.
Artifact: retro.md.

2.4.2 Communication & Reporting

□ All critical tests are green in CI
□ Ethics sign-off completed for new data or model changes
□ QA acceptance criteria met
□ Security scan completed with no critical findings
□ Deployment runbook updated
□ Rollback tested
□ Stakeholders notified

Save as release-checklist|md.

Handoffs and artefacts for each stage

- From Tech Lead to QA: architecture doc, API spec, testable demo link, test data.
- From QA to PM: test report, bug list, release recommendation.
- From Ethics to PM/Tech Lead: DPIA, model card, mitigation plan.
- From PM to Stakeholders: release notes, impact summary

Store everything in a single project drive with versioning and clear filenames.

2.4.3 First 30/60/90 day plan (high level)

Days 0–30: hire core roles, set up repos, draft architecture, ethics DPIA kickoff, basic CI, first sprint.
Days 31–60: build MVP features, automated tests, initial model prototype, internal ethics review, first internal release.
Days 61–90: public pilot, monitor metrics, close ethics and QA actions, iterate on feedback.

2.4.4 Escalation path and backups

If blocker > 48 hours: escalate to PM.

If PM cannot resolve within 24 hours: escalate to Steering Committee members (list names or [TBD]).
If production incident: follow runbook and notify Security and Ethics leads within 2 hours.
Maintain an escalation-protocol.md.

2.4.5 Documentation, templates and filenames to create now

- project-plan.md
- architecture.md
- api-spec.yaml
- data-inventory.csv
- ethics-checklist.md
- model-card.md
- test-plan.md
- release-checklist.md
- weekly-status.md
- risks.csv

Quick governance rules to adopt today

1. Every release requires ethics sign-off if model or data changes.
2. Pull requests must have at least one review from Tech Lead and one from QA for core services.
3. Any dataset containing personal data must be logged in data-inventory.csv before use.
4. Security findings of severity high must block release.

Example reporting templates (ready to copy)

Weekly status report (short)

- Sprint: [number]
- Progress: what shipped this week
- Blockers: top 3 issues and owner
- Risks: new or escalated risks
- Next steps: priorities for next week

Release note (short)

- Version:
- Date:
- Summary: one-line summary
- Changes: bullet list of features or fixes
- Known issues: brief
- Rollback plan: one line

Next actions for you

1. Copy the Team_Roster.xlsx template and add current people or [TBD].
2. Create the repo and add the documentation files listed in section 12.
3. Run a 45-minute kick-off with the four leads to confirm the 30/60/90 plan and RACI.
4. Assign one person to own recruitment for each [TBD] role.

2.4.6 Risk Management

- **Key Risks:**
  - Data availability gaps
  - Platform API restrictions
  - Model bias & fairness issues
  - Security breaches (targeting by malicious actors)
  - Misuse of outputs by third parties

- **Mitigation Plans:**
  - Redundancy in data sources
  - Human oversight in model outputs
  - Secure comms & infra monitoring
  - Clear ethics guardrails

Chapter 03 Project Implementation

3.1 Phase 1: Project Definition & Planning

3.1.1 Define Project Scope, Objectives & Success Metrics

Action: Write a Project Scope Statement (problem definition, target users, outcomes).

Action: List objectives (e.g., monitor election narratives across platforms, produce weekly briefs, test ML for narrative extraction).

Action: Define success metrics:

- Coverage: % of narratives captured on TikTok, Telegram, X, etc.
- Outputs: # of reports, dashboards, briefs published.
- Performance: Latency of queries, ML model accuracy.

Deliverable: project-scope.md

Checklist:

- Context documented (actors, narratives, timelines).
- Goals written (outputs, reports, dashboard features).
- KPIs defined and measurable.

3.1.2 Draft Comprehensive Project Plan

Action: Break down the project into phases → *Planning* → *Data* → *Build* → *Test* → *Deploy* → *Monitor*.

Action: Create a workplan (Gantt or Kanban). Include milestones, owners, deadlines, dependencies.

Action: List resources: people, infrastructure, budget, tools.

How to create a tech workplan:

a. Translate objectives → epics (e.g., “Election data pipeline”).
b. Break epics → tasks (e.g., “Scraper for TikTok hashtags”).
c. Assign roles (PM, Tech Lead, QA).
d. Sequence tasks in time blocks (2–3 week sprints).
e. Add dependencies (e.g., *scraper* must exist before *NLP pipeline*).
f. Deliverable: project-plan.xlsx (with Gantt chart or sprint board).

Establish Communication Protocols

3.1.3 Establish Communication Protocols

Action: Define main channels for internal comms:
- Slack/Teams/Signal → daily operations.
- WhatsApp → quick alerts/emergency comms.
- Discord → dev coordination (esp. for bots).
- Email → formal updates/stakeholders.

Action: Define reporting cadence:
- Weekly → progress + blockers.
- Monthly → stakeholder reports + metrics.
- Post-release → retro notes.

Deliverable: comms-protocol.md

3.1.4 Conduct Risk Assessment & Mitigation

Risks to check:
- *Technical* → scraper breaks, API changes.
- *Data* → low coverage, noisy or biased.
- *Security* → leaks, DDoS, account takeover.
- *Reputational* → ethical controversy, misinterpretation of outputs.

Action: List risks in risk-register.csv with severity + mitigation.

3.1.5 Define Data Requirements

Action: List platforms (TikTok, Telegram, X, Facebook, etc.).

Action: Define formats (video, text, images, metadata).

Action: Specify languages/dialects to cover.

Action: Integrate your TikTok & Telegram "how to do it" doc as an annex here. It becomes the standard operating procedure (SOP) for pulling data from these platforms.

Deliverable: data-requirements.md + SOP annex.

3.1.6 Identify Initial Data Sources & Feasibility

Action: Map each platform:
- Can you use an API?
- Do you need scraping?
- Do you need a partnership (e.g., fact-checkers, NGOs)?

Deliverable: data-sources.xlsx

3.1.7 Draft Initial Ethical Guidelines

Action: Write a one-pager on fairness, transparency, review/appeal mechanisms.

Action: Flag early concerns:
- Bias in data sampling (e.g., only urban TikTok).
- Privacy of Telegram users.

Deliverable: ethics-guidelines.md

3.1.8 Flag Data Quality Issues

Checklist:

- Duplication across platforms?
- Gaps in coverage?
- Metadata completeness?
- Bias risks?

Document in data-quality-report.md.

3.1.9 Gather ML Requirements

Tasks:

- Identify NLP tasks (NER for actors, RE for relationships, topic clustering for narratives).
- Research candidate ML tools/frameworks (spaCy, Hugging Face, custom models).
- Estimate effort + compute cost.

Deliverable: ml-requirements.md

3.1.10 Start Technical Architecture Discussions

Action: Define storage (SQL, NoSQL, blob).
Action: Define APIs needed for data + dashboard.
Action: Plan for scalability (cloud or on-prem).
Deliverable: architecture-draft.md

3.1.11 Checklist:

- Repositories (GitHub/GitLab).
- Version control workflows (branching strategy).
- CI/CD pipelines.
- Dev environments (Docker, venv).

Deliverable: dev-setup.md

3.1.12 Gather UI/UX Requirements

Action: Interview target users (journalists, civil society, analysts).
Action: Document workflows + pain points.
Action: Define what they need to *see/do* in dashboards.
Deliverable: ux-requirements.md

3.1.13 Research Visualization Tools

Options: D3.js, vis.js, Plotly, Observable, PowerBI.
Deliverable: viz-research.md

3.1.14 Draft Initial Wireframes & Mockups

Action: Create low-fidelity sketches (Figma, Miro, even paper).
Deliverable: wireframes.pdf

3.1.15 Define Testing Scope Early

Action: List key test scenarios:

- Data ingestion reliability.
- NLP accuracy (F1, precision, recall).
- Dashboard performance.
- User acceptance tests.

Deliverable: testing-scope.md

3.1.16 End of Phase 1 Output Package 📌

□ project-scope.md
□ project-plan.xlsx
□ comms-protocol.md
□ risk-register.csv
□ data-requirements.md (with TikTok/Telegram SOP annexed)
□ data-sources.xlsx
□ ethics-guidelines.md
□ data-quality-report.md
□ ml-requirements.md
□ architecture-draft.md
□ dev-setup.md
□ ux-requirements.md
□ viz-research.md
□ wireframes.pdf
□ testing-scope.md

3.2 Phase 2: Data Acquisition & Pre-Processing

This phase focuses on setting up pipelines to capture raw data, clean it, and prepare it for ML + analysis.

3.2.1 Build Acquisition Pipelines

Action: For each platform, decide method:

- API (if available).
- Scrapers (custom Python/Node).
- Database connectors (for structured feeds).

3.2.1.1 How to Scrape TikTok with TikTok Scraper?

TikTok Scraper is user-friendly, offering a smooth start even for those who have never extracted data from TikTok before.

Here's how to scrape TikTok with TikTok Scraper in some few steps:

i. Create a free Apify account using your email.
ii. Open TikTok Scraper.
iii. Add one or more hashtags to scrape TikTok videos and creators.
iv. Click the "Start" button and wait for the data to be extracted.
v. Download your data in JSON, XML, CSV, Excel, or HTML.
vi. Choose the fields, for example: webVideoUrl, authorMeta/name, detailedMentions/1/name, authorMeta/nickName, createTimeISO, and text.

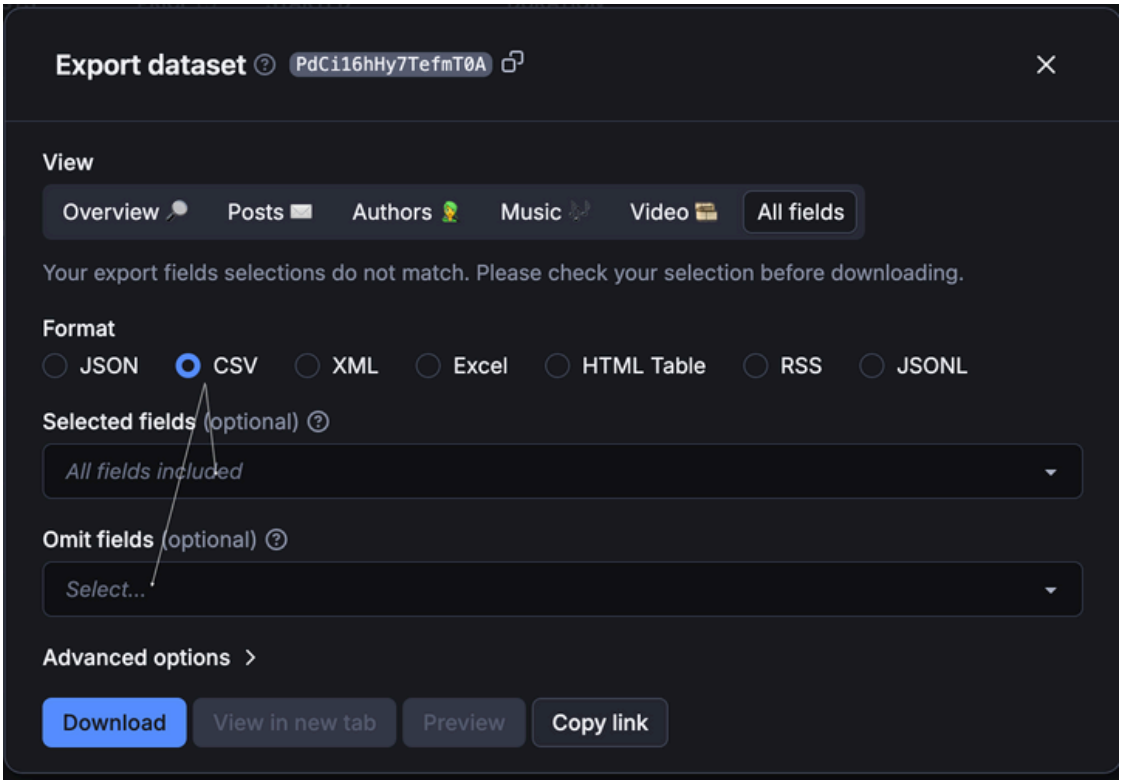


Fig : How to Scrape TikTok with TikTok Scraper?

- Data extracted in this format can be uploaded

For more information, read this step-by-step guide or watch our short video tutorial. It also applies to TikTok Data Extractor.
Once data has been downloaded, delete all other fields and retain the following fields.

	A	B	C	D	E	F	G	H	I	J	K
1	authorMeta/id	authorMeta/name	authorMeta/nickName	createTime	createTimeISO	diggCount	input	mentions/0	searchHashta	text	webVideoUrl
2	7443105084555	aparoblog	APARO-BLOG	1751950235	2025-07-08T04:50:3	50	#tinubu		tinubu	President Bola A	https://www.tikto
3	7470607018627	amos.vlogs3	Amos Vlogs	1750928119	2025-06-26T08:55:1	32400	#tinubu		tinubu	I wish there were	https://www.tikto
4	7106485554611	dr_ada00	Dr Ada Nwosu	1751894579	2025-07-07T13:22:5	3072	#tinubu		tinubu	This student loar	https://www.tikto
5	7463955346722	mynaijaplus	mynaijaplus	1751895242	2025-07-07T13:34:0	2471	#tinubu		tinubu	President Tinubu	https://www.tikto
6	7036587690623	energygoddexx	mimi_yakigar	1751874020	2025-07-07T07:40:2	832	#tinubu		tinubu	Under Tinubu, N	https://www.tikto
7	7376642646020	iakodend	Ibrahim Aminu Adam	1751796689	2025-07-06T10:11:2	224	#tinubu		tinubu	#news #nigeria #	https://www.tikto
8	7129924436644	primateayodele	Primate Ayodele	1751825614	2025-07-06T18:13:3	1966	#tinubu		tinubu	#iescworldwide #	https://www.tikto
9	6806417717697	lagosbahrainlondon	Loko~Of~London	1751824649	2025-07-06T17:57:2	550	#tinubu		tinubu	#goodnews #trul	https://www.tikto
10	7419604144850	thegeneral0012	TheGeneral	1751810322	2025-07-06T13:58:4	8048	#tinubu	@ÅŸØRÅÑGÊ	tinubu	We are a poor n	https://www.tikto
11	7073980339049	renosmasterclass	Reno Omokri	1751840266	2025-07-06T22:17:4	342	#tinubu		tinubu	#politics #africa	https://www.tikto
12	6887775049982	gpgaines	Gp	1741238221	2025-03-06T05:17:0	292600	#tinubu		tinubu	Aura Dey cryyyy	https://www.tikto
13	7151460002631	bestcapcutedits01	Best Capcut Edits	1751816934	2025-07-06T15:48:5	1877	#tinubu		tinubu	#amapiano #chr	https://www.tikto
14	7219810135969	therealisttv1	The Realist TV	1751810817	2025-07-06T14:06:5	5171	#tinubu		tinubu	Peter Obi if you	https://www.tikto
15	7475274036232	inspectorlunge_	inspectorlunge_	1741970980	2025-03-14T16:49:4	2900000	#tinubu		tinubu	#presidenttinubu	https://www.tikto
16	7092417947083	.iripsman	IRIPSMAN	1751446702	2025-07-02T08:58:2	3570	#tinubu		tinubu	facts #video	https://www.tikto
17	6941403635750	kashim_shettimatv	KASHIM SHETTIMA	1751560131	2025-07-03T16:28:5	1053	#tinubu		tinubu	#kashimshettima	https://www.tikto
18	7470607018627	amos.vlogs3	Amos Vlogs	1751284224	2025-06-30T11:50:2	235600	#tinubu		tinubu	"We eat the skin	https://www.tikto
19	7343698638365	soskideditz	Soskid Editz	1722805307	2024-08-04T21:01:4	41900	#tinubu		tinubu	Tinubu	https://www.tikto
20	7069885385199	omobaba183	Tinubu	1685369982	2023-05-29T14:19:4	22	#tinubu		tinubu	#tinubuforpresid	https://www.tikto
21	7443105084555	aparoblog	APARO-BLOG	1750830808	2025-06-25T05:53:2	49500	#tinubu		tinubu	Steeze , Com	https://www.tikto
22	7470607018627	amos.vlogs3	Amos Vlogs	1751055621	2025-06-27T20:20:2	32900	#tinubu		tinubu	A common screv	https://www.tikto
23	7374192723093	ghanamemes	Ghana Memes	1742466693	2025-03-20T10:31:3	111700	#tinubu	@Ghana Mem	tinubu	Make a enjoy sh	https://www.tikto
24	7348006478504	nigeria.liberatio	Nigeria Liberation Moveme	1751579061	2025-07-03T21:44:2	1181	#tinubu		tinubu	Yoruba INTERES	https://www.tikto
25	6763540298486	eveningstandard	The Standard	1677861741	2023-03-03T16:42:2	35000	#tinubu		tinubu	Bola Tinubu, hac	https://www.tikto
26	6816338224891	i.mjoana	i.mjoana	1750524886	2025-06-21T16:54:4	798	#tinubu		tinubu	#nigeria #tinubu	https://www.tikto
27	7069885385199	omobaba183	Tinubu	1685370475	2023-05-29T14:27:5	43	#tinubu		tinubu	#tinubu2023	https://www.tikto
28	7482590884383	jhh2768	jhh	1742511943	2025-03-20T23:05:4	29700	#tinubu		tinubu	Anticolonialism	https://www.tikto
29	7146994269110	red_devi11	Tinubu	1751280980	2025-06-30T10:56:2	58	#tinubu		tinubu	#codm	https://www.tikto

The data is ready to be uploaded for narrative analysis on the text and timelines, as well as network analysis on the accounts.

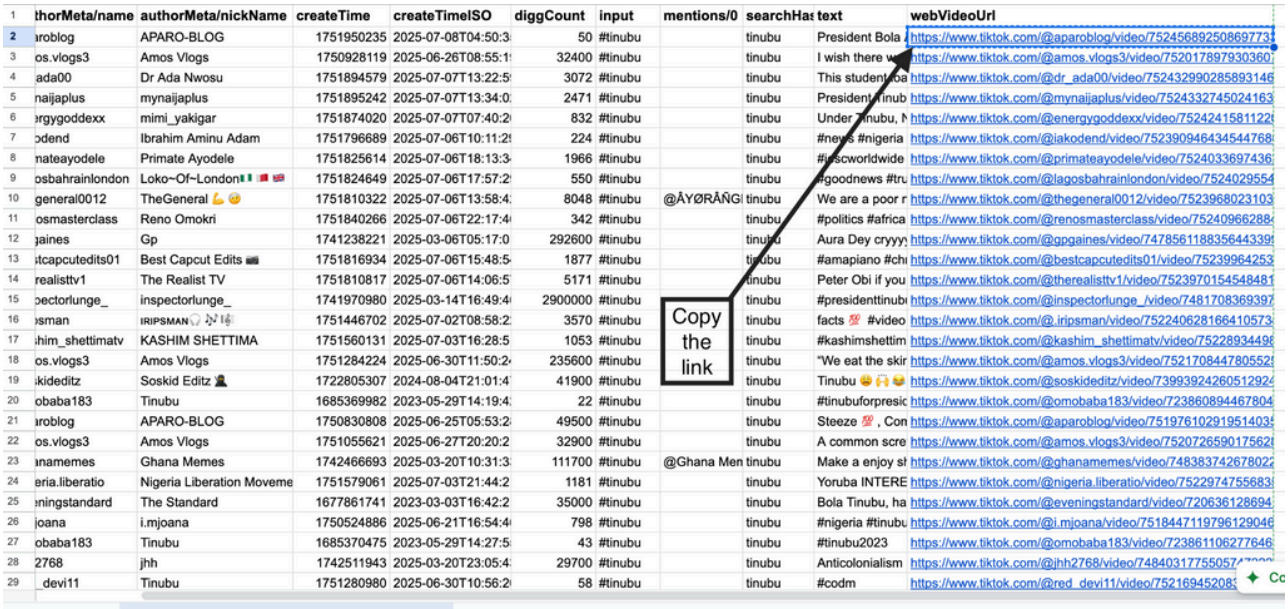
3.2.1.2 TikTok Video Transcription

There is no tool that allows searching based on transcription of sounds/overlayed text; however, one can transcribe individual videos if one has a small sample of data.

Step for Transcribing individual videos.

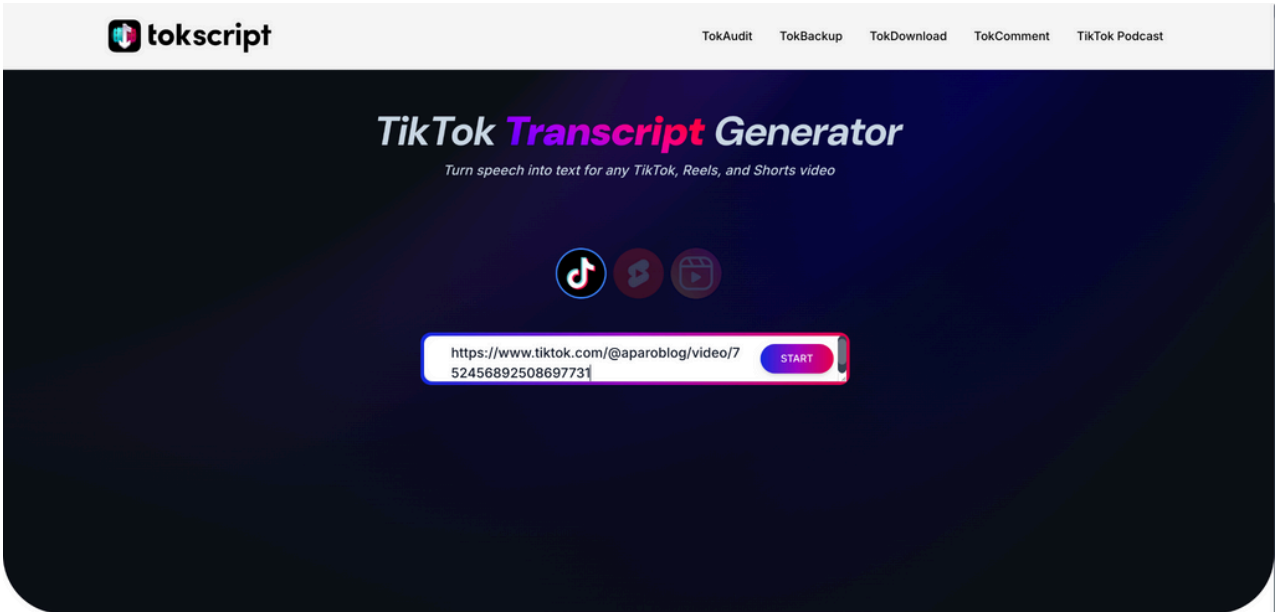
a. Download your data in JSON, XML, CSV, Excel, or HTML from Apify.

b. Once you have your data, copy the Video URL.



	authorMeta/name	authorMeta/nickName	createTime	createTimeISO	diggCount	input	mentions/0	searchHas	text	webVideoUrl
2	aroblog	APARO-BLOG	1751950235	2025-07-08T04:50:3	50	#tinubu		tinubu	President Bola	https://www.tiktok.com/@aparoblog/video/7524568925086977
3	os.vlogs3	Amos Vlogs	1750928119	2025-06-26T08:55:1	32400	#tinubu		tinubu	I wish there w	https://www.tiktok.com/@amos.vlogs3/video/7520178979303607
4	ada00	Dr Ada Nwosu	1751894579	2025-07-07T13:22:5	3072	#tinubu		tinubu	This student ba	https://www.tiktok.com/@dr_ada00/video/7524329902858931
5	naijaplus	mynaijaplus	1751895242	2025-07-07T13:34:0	2471	#tinubu		tinubu	President Tinub	https://www.tiktok.com/@mynaijaplus/video/7524332745024163
6	rgygoddexx	mimi_yakigar	1751874020	2025-07-07T07:40:2	832	#tinubu		tinubu	Under Tinubu, N	https://www.tiktok.com/@energygoddexx/video/7524241581122
7	odend	Ibrahim Aminu Adam	1751796689	2025-07-06T10:11:2	224	#tinubu		tinubu	#news #nigeria	https://www.tiktok.com/@iakodend/video/7523909464354476
8	nateayodele	Primate Ayodele	1751825614	2025-07-06T18:13:3	1966	#tinubu		tinubu	#bscworldwide	https://www.tiktok.com/@primateayodele/video/7524033697436
9	osbahrainlondon	Loko~Of~London	1751824649	2025-07-06T17:57:2	550	#tinubu		tinubu	#goodnews #tru	https://www.tiktok.com/@lagosbahrainlondon/video/7524029554
10	general0012	TheGeneral	1751810322	2025-07-06T13:58:4	8048	#tinubu	@ÀŸØRÅÑG	tinubu	We are a poor r	https://www.tiktok.com/@thegeneral0012/video/7523968023103
11	osmasterclass	Reno Omokri	1751840266	2025-07-06T22:17:4	342	#tinubu		tinubu	#politics #africa	https://www.tiktok.com/@renosmasterclass/video/7524096628
12	gaines	Gp	1741238221	2025-03-06T05:17:0	292600	#tinubu		tinubu	Aura Dey cryyy	https://www.tiktok.com/@gpgaines/video/7478561188356443395
13	tcapcutedits01	Best Capcut Edits	1751816934	2025-07-06T15:48:5	1877	#tinubu		tinubu	#amapiano #chi	https://www.tiktok.com/@bestcapcutedits01/video/7523996425
14	realisttv1	The Realist TV	1751810817	2025-07-06T14:06:5	5171	#tinubu		tinubu	Peter Obi if you	https://www.tiktok.com/@therealisttv1/video/7523970154548481
15	pectorlunge_	inspectorlunge_	1741970980	2025-03-14T16:49:4	2900000	#tinubu		tinubu	#presidenttinubu	https://www.tiktok.com/@inspectorlunge_/video/7481708369397
16	sman	IRIPSMAN	1751446702	2025-07-02T08:58:2	3570	#tinubu		tinubu	facts #video	https://www.tiktok.com/@.iripsman/video/7522406281664105
17	him_shettimatv	KASHIM SHETTIMA	1751560131	2025-07-03T16:28:5	1053	#tinubu		tinubu	#kashimshettim	https://www.tiktok.com/@kashim_shettimatv/video/7522893449
18	os.vlogs3	Amos Vlogs	1751284224	2025-06-30T11:50:2	235600	#tinubu		tinubu	"We eat the skir	https://www.tiktok.com/@amos.vlogs3/video/7521708447805525
19	kideditz	Soskid Editz	1722805307	2024-08-04T21:01:4	41900	#tinubu		tinubu	Tinubu	https://www.tiktok.com/@soskideditz/video/7399392426051292
20	obaba183	Tinubu	1685369982	2023-05-29T14:19:4	22	#tinubu		tinubu	#tinubuforpresic	https://www.tiktok.com/@omobaba183/video/7238608944678044
21	aroblog	APARO-BLOG	1750830808	2025-06-25T05:53:2	49500	#tinubu		tinubu	Steeze , Con	https://www.tiktok.com/@aparoblog/video/7519761029195140
22	os.vlogs3	Amos Vlogs	1751055621	2025-06-27T20:20:2	32900	#tinubu		tinubu	A common scre	https://www.tiktok.com/@amos.vlogs3/video/7520726590175628
23	namemes	Ghana Memes	1742466693	2025-03-20T10:31:3	111700	#tinubu	@Ghana Men	tinubu	Make a enjoy sh	https://www.tiktok.com/@ghanamemes/video/7483837426780224
24	eria.liberatio	Nigeria Liberation Moveme	1751579061	2025-07-03T21:44:2	1181	#tinubu		tinubu	Yoruba INTERE	https://www.tiktok.com/@nigeria.liberatio/video/7522974755683
25	eningstandard	The Standard	1677861741	2023-03-03T16:42:2	35000	#tinubu		tinubu	Bola Tinubu, ha	https://www.tiktok.com/@eveningstandard/video/7206361286947
26	joana	i.mjoana	1750524886	2025-06-21T16:54:4	798	#tinubu		tinubu	#nigeria #tinubu	https://www.tiktok.com/@i.mjoana/video/7518447119796129046
27	obaba183	Tinubu	1685370475	2023-05-29T14:27:5	43	#tinubu		tinubu	#tinubu2023	https://www.tiktok.com/@omobaba183/video/7238611062776467
28	2768	jhh	1742511943	2025-03-20T23:05:4	29700	#tinubu		tinubu	Anticolonialism	https://www.tiktok.com/@jhh2768/video/7484031775505
29	_devi11	Tinubu	1751280980	2025-06-30T10:56:2	58	#tinubu		tinubu	#codm	https://www.tiktok.com/@red_devi11/video/7521694520

c. Paste the URL into the TokScript Free Video Transcript Generator.

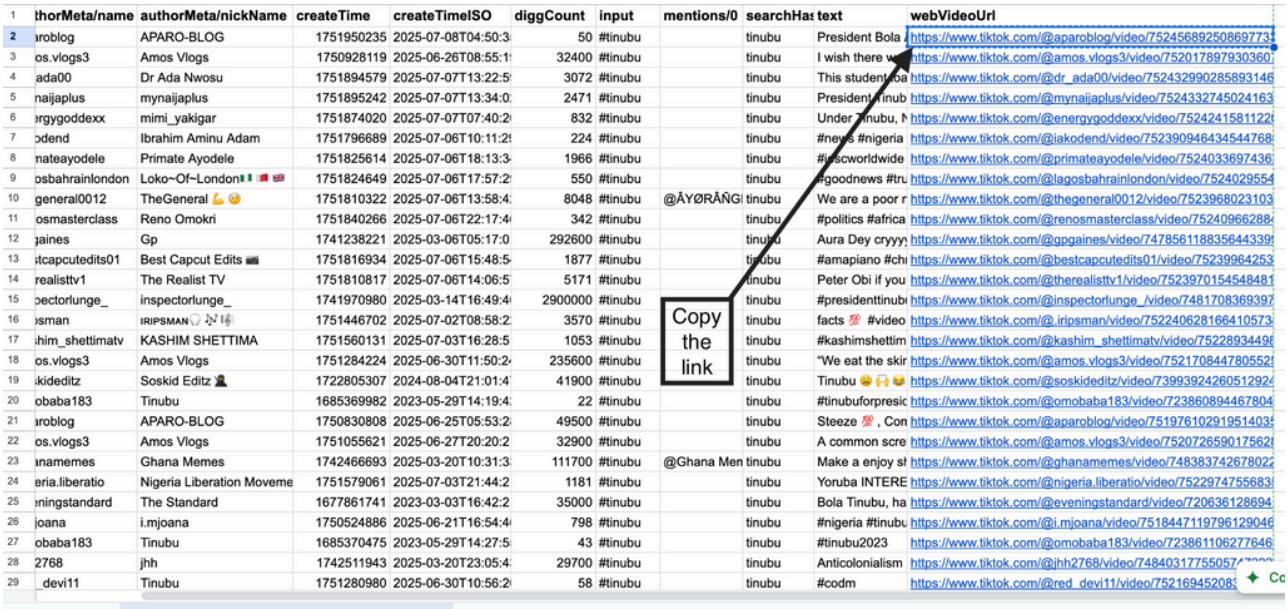


d. Click Start to generate the transcript.

e. Once the transcript has been generated, copy the text and remove the timestamps.

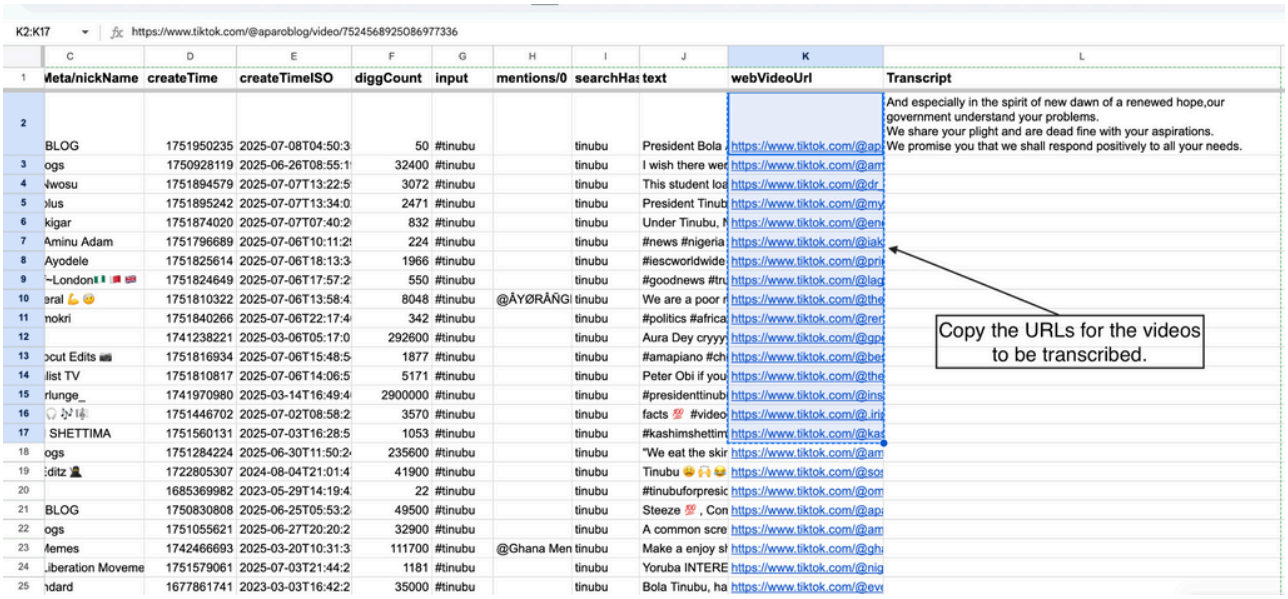
f. In your dataset, create a new column for the transcript.

g.Once you are done with the videos transcription Upload them to the election watch for analysis.

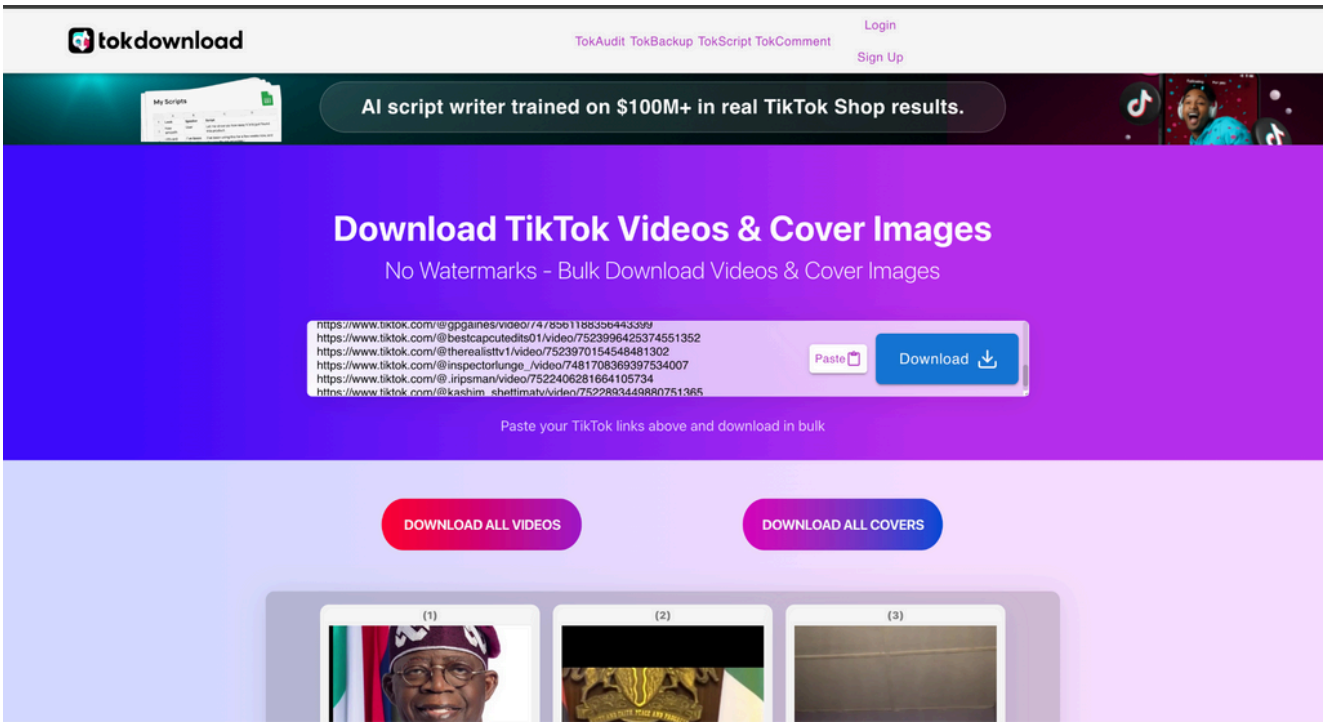


Steps for downloading the mass videos

a. Download your data in JSON, XML, CSV, Excel, or HTML from Apify.
b. Copy the the Url for all the video you need to down



c.Post the links on Tokdownload tool.



c.Post the links on Tokdownload tool.
d.Once the videos are ready download them.
e.Upload your videos to your tool of choice for transcription.

3.2.1.3 Script Overview for automating this process

This script automates the downloading and transcription of TikTok videos using:

yt_dlp for video downloading, whisper (OpenAI's ASR model) for transcription, pandas for saving results into a CSV.

Code Breakdown:

a. Imports and Dependencies

```
import os
import whisper
import yt_dlp
import pandas as pd
from pathlib import Path
```

os: For file system operations.
whisper: For automatic speech recognition (ASR).
yt_dlp: For downloading videos from TikTok.
pandas: For handling tabular data.
Pathlib: (Imported but unused in this code.)

a. Configuration

```
# ==== SETTINGS ====
COOKIES_FILE = "tiktok.com_cookies.txt"
LINKS_FILE = "links.csv"
OUTPUT_DIR = "downloads"
MODEL_SIZE = "turbo"  # whisper model: tiny, base, small, medium, large
```

Defines:
Path to a cookie file for TikTok authentication.
CSV file containing TikTok URLs to download.
Output directory for downloaded videos and transcripts.
Whisper model size to use (e.g., turbo for speed).

c.Setup

```
# ==== SETUP ====
os.makedirs(OUTPUT_DIR, exist_ok=True)
model = whisper.load_model(MODEL_SIZE)
transcriptions = []
```

Ensures the output directory exists.
Loads the selected Whisper model.
Initializes a list to store results.

d.Load TikTok URLs

```
# ==== LOAD LINKS ====
with open(LINKS_FILE, "r") as f:
    links = [line.strip() for line in f if line.strip()]
```

Read and filter non-empty TikTok links from the links.csv file.

e.Process Each Video
Loop through each link and perform:

- Download Video

```
# ==== DOWNLOAD + TRANSCRIBE ====
for i, url in enumerate(links, 1):
    print(f"({i}/{len(links)}) Downloading: {url}")
    try:
        ydl_opts = {
            'outtmpl': f'{OUTPUT_DIR}/video_{i}.mp4',
            'format': 'mp4',
            'cookiefile': COOKIES_FILE,
            'quiet': True,
            'noplaylist': True
        }

        with yt_dlp.YoutubeDL(ydl_opts) as ydl:
            ydl.download([url])
        video_path = f'{OUTPUT_DIR}/video_{i}.mp4'
```

Uses `yt_dlp` to download the video using cookie authentication and saves it in `OUTPUT_DIR`

- Transcribe Audio

```
        print(f"→ Transcribing: {video_path}")
        result = model.transcribe(video_path)
        transcription = result["text"]

        transcriptions.append({
            "url": url,
            "filename": f"video_{i}.mp4",
            "transcription": transcription
        })

        with open(f"{OUTPUT_DIR}/video_{i}.txt", "w") as f:
            f.write(transcription)

    except Exception as e:
        print(f"Failed on {url}: {e}")
        transcriptions.append({
            "url": url,
            "filename": f"video_{i}.mp4",
            "transcription": "[ERROR]"
        })
```

Transcribes the audio using Whisper and stores the text. Saves the transcription to a `.txt` file.
Also appends metadata and transcription to a list.
Error Handling: Logs any errors and records a placeholder transcription.

f.Save All Results

```
# ==== SAVE ALL TO CSV ====
df = pd.DataFrame(transcriptions)
df.to_csv("transcriptions.csv", index=False)
print("Done. Transcriptions saved to transcriptions.csv")
```

Converts transcription metadata into a DataFrame. Saves it to `transcriptions.csv`.

Observations
The script assumes the presence of `links.csv` and `tiktok.com_cookies.txt`.
No multithreading or async; processes videos sequentially.

3.2.1.4 Telegram channel data scraping

This section outlines the steps followed to identify, locate and extract data from Telegram channels using Junkipedia, with the aim of supporting an investigation into disinformation and election-related narratives in Nigeria and beyond.

a.Source identification

The initial source of potential Telegram channels was the file titled Telegram data located in the Data From 18 Sources folder.

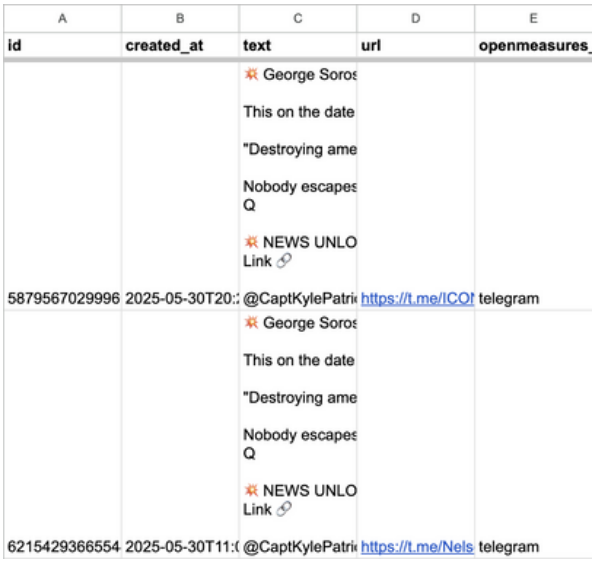
From this dataset, the context_name column was used as the primary identifier for each Telegram channel.

H	I	J	K	L	M	N	O	P	Q
engagement_re	engagement_di	engagement_re	engagement_vi	openmeasures_	openmeasures_	openmeasures_	openmeasures_	context_nan	actor_usernam
1		1	118					ICONS(INTERN	Jules FightLik
			22					NELSON PEOPLE AWAKE	

b.Channel verification & URL discovery

For each entry:

- Copy the context_name value and manually search for the Telegram channel online to verify its existence.
- Once identified, navigate to specific post URLs within the dataset, which are often linked directly to the original Telegram messages.
- From these URLs, you can directly extract both the channel name and URL from Telegram, which is more reliable than relying solely on metadata from the original file.

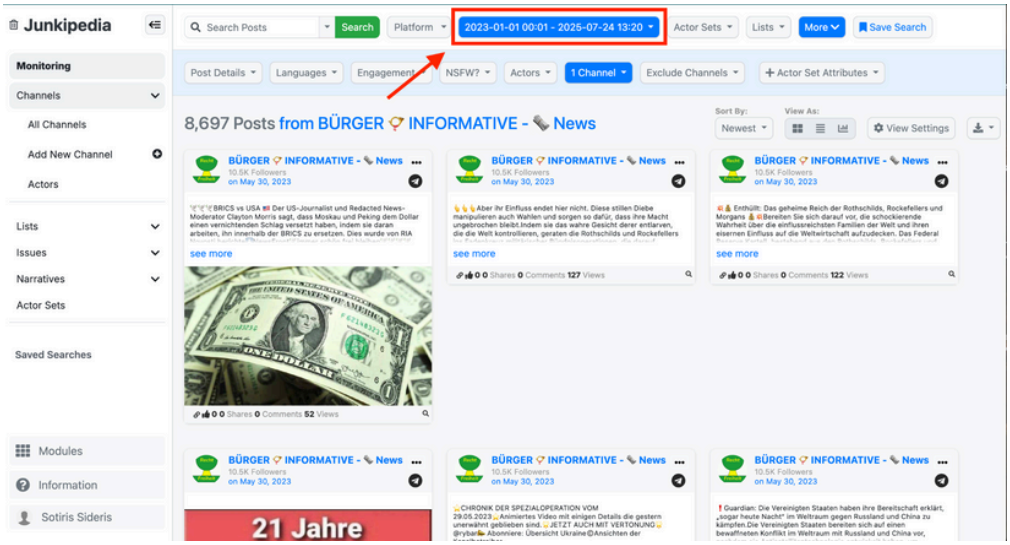


A	B	C	D	E
id	created_at	text	url	openmeasures_
5879567029996	2025-05-30T20::	George Soros This on the date "Destroying ame Nobody escapes Q NEWS UNLO Link	@CaptKylePatri	https://t.me/ICOI telegram
6215429366554	2025-05-30T11:(	George Soros This on the date "Destroying ame Nobody escapes Q NEWS UNLO Link	@CaptKylePatri	https://t.me/Nels telegram

c.Integration with Junkipedia

After obtaining the verified channel names and URLs:

- Search within Junkipedia to determine whether each channel was already being monitored by someone else.
- If the channel was found:
  - Schedule data retrieval starting from 01/01/2023 to present



So far, we have added data from 11 Telegram channels to this folder, with more being added as they come from Junkipedia.

These include both newly monitored channels and those already tracked by others.

e.Data characteristics

The exported CSVs contain structured information, with key columns including:

Field	Description
PostId	Unique identifier of each post
PostUrl	Direct link to the post on Telegram
PostEngagement	Blank or legacy engagement metadata
Platform	Always set to "Telegram"
ChannelName, ChannelID, ChannelUrl	Channel identifiers
post_body_text	The visible text content of the post
published_at	UTC timestamp of publication
post_media_urls	Link(s) to image/video media files
LikesCount, SharesCount, CommentsCount, ViewsCount	Engagement stats (often zeroed out or null)
EmbeddedContentText	Any links or quotes from other platforms (e.g., X/Twitter)
post_data	JSON blob with follower/view counts at time of retrieval

Example post texts range from QAnon content and conspiracy theory claims to general political commentary and U.S.-centric misinformation.

Observations

- Many high-volume channels yielded thousands of posts for the given timeframe.
- Although not all content is Nigeria-specific, the disinformation style and influencer patterns remain consistent with our broader comparative analysis.
- The current dataset is sufficiently rich for early analysis and additional channels will enhance diversity and regional relevance.

Notes for analysis team

- Ensure the Telegram data can be analyzed using the same NLP pipelines or scripts used for Twitter data.
- Be aware that some post_body_text fields contain raw HTML or embedded media links that may require cleaning.
- Missing fields (e.g., engagement counts) are expected and not necessarily errors – this reflects Telegram's limited analytics.

Deliverable: acquisition-pipelines.md with platform-specific sections.

3.2.2 Clean & Normalise Data

Action: Standardise formats (e.g., timestamps → UTC ISO8601).

Action: Deduplicate content across channels (e.g., same TikTok reposted in Telegram).
Action: Convert encodings (UTF-8).
Deliverable: data-cleaning-scripts/ repo + cleaning-log.md.

3.2.3 Handle Missing Values & Outliers

Action: Define rules:

- Missing text → discard or impute.
- Outlier posts (e.g., sudden 1M views) → flag for QA review.

Deliverable: missing-values-policy.md.

3.2.4 Perform Feature Engineering

Tasks:

- Extract metadata (author, channel, hashtags).
- Convert text → embeddings (e.g., Sentence-BERT).
- Compute TF-IDF for keyword analysis.
- Add language ID and sentiment tags.

Deliverable: feature-engineering.md.

3.2.5 Document Schema & Maintain Data Dictionary

Action: Create schema (tables/collections: posts, users, metadata).
Action: Build a data-dictionary.xlsx describing every field, type, source.
Deliverable: schema.md + data-dictionary.xlsx.

3.2.6 Set Up Secure Ingestion Services & APIs

Action: Wrap scrapers/collectors in a microservice or cron job.
Action: Use secure credentials storage (Vault, .env, Secrets Manager).
Deliverable: ingestion-services/ repo.

3.2.7 Configure Data Storage

Options:

- SQL DB (Postgres/MySQL) for structured.
- NoSQL (MongoDB, ElasticSearch) for unstructured.
- Data Lake (S3/GCS) for raw dumps.

Deliverable: storage-config.md.

3.2.8 Implement Data Validation Checks

Action: Validate at ingestion:

- Field presence.
- Correct formats (timestamps, URLs).
- Language detection matches expected set.

Deliverable: validation-scripts/ + validation-log.md.

3.2.9 Develop Ingestion Test Cases

Action: Write unit tests for scrapers/parsers.
Action: Write QA tests for daily batch ingestion.
Deliverable: tests/ingestion-tests.py.

3.2.10 Perform Data Integrity Testing

Checks:

- Record counts per day vs expected volume.
- Compare scraped vs API totals.
- Spot-check duplicates.

Deliverable: integrity-report.md.

3.2.11 Prepare Labelled Datasets

- Action: For supervised ML tasks (NER, classification):
  - Define annotation guidelines.
  - Use tools like Prodigy, Label Studio, or doccano.
  - Pilot with a small dataset → refine.
- Deliverable: labelled-datasets/ repo + annotation-guide.md.

End of Phase 2 Output Package

□ acquisition-pipelines.md (with TikTok/Telegram SOP annex)
□ data-cleaning-scripts/ + cleaning-log.md
□ missing-values-policy.md
□ feature-engineering.md
□ schema.md + data-dictionary.xlsxv
□ ingestion-services/ repo
□ storage-config.md
□ validation-scripts/ + validation-log.md
□ tests/ingestion-tests.py
□ integrity-report.md
□ labelled-datasets/ + annotation-guide.md

3. 3 Phase 3: ML Model Development & Training

Goal: Build and train machine learning components (narrative extraction, NER, RE) using the multi-agent system framework (ml/) to support misinformation tracking, actor mapping, and trend analysis.

3.3.1 System Preparation

Steps

a. Set up Python environment:

```
cd ml
python3 -m venv .venv
source .venv/bin/activate
pip install -r requirements.txt
gcloud auth application-default login
```

b. Configure environment variables:
   - GOOGLE_CLOUD_PROJECT -GCP project ID
   - GOOGLE_CLOUD_LOCATION -GCP region
   - MONGODB_ATLAS_URI -MongoDB connection string
   - *(Optional)* OPEN_ROUTER_API_KEY - for AI report formatting

Verify CoordinatorAgent and specialist agents (DataEngAgent, OsintAgent, LexiconAgent, TrendAnalysisAgent) are discoverable and runnable using:`adk run ml.main:coordinator_agent`

Deliverables:

- dev-setup.md (document setup steps and environment variables)
- .env file (not committed to git)

3.3.2 Narrative Extraction

Steps

a. Ingest cleaned text data from TikTok/Telegram pipelines.
b. Run topic modelling experiments:
   - LDA → baseline topic structure
   - BERTopic → contextual clustering for multilingual posts
c. Implement summarisation pipelines:
   - TextRank (extractive)
   - Pegasus/T5 (abstractive, fine-tuned for short posts)
d. Evaluate coherence, diversity, and cluster stability.

Deliverables:

- narrative-experiments.md
- narrative-model/ (saved model artefacts)

3.3.3 Named Entity Recognition (NER)

Steps

a. Annotate sample election data with entities (people, orgs, places, events).
b. Fine-tune a multilingual transformer model (e.g. XLM-RoBERTa or spaCy transformer) on this dataset.
c. Add entity normalisation layer (map aliases to canonical names).
d. Evaluate using precision, recall, and F1 per entity type.

Deliverables:

- ner-model/
- ner-eval-report.md

3.3.4 Relationship Extraction (RE)

Steps

a. Define relationship schema (e.g. *person-funded-event*, *org-affiliated-with-org*).
b. Build RE system using:
    - Dependency parsing + rules for simple patterns
    - Transformer-based supervised RE for complex links
c. Link output entities from NER to construct a knowledge graph.

Deliverables:

- re-model/
- relationship-schema.md
- relationship-sample-graph.json

3.3.5 Training Pipeline

Steps

a. Split data: 70% train, 15% dev, 15% test.
b. Maintain stratified sampling by platform and language.
c. Track all experiments in MLflow or W&B.
d. Save configs and hyperparameters in YAML for reproducibility.

Deliverables:

- training-datasets/
- experiment-tracker/

3.3.6 Evaluation & Fairness

Steps

a. Evaluate models using:
    - Accuracy, precision, recall, F1
    - Per-language performance breakdown
b. Conduct bias audit:
    - Compare errors by region, language, actor category
    - Document fairness mitigations (re-weighting, data augmentation)
c. Sign off with Ethics & Governance Lead before integration.

Deliverables:

- evaluation-report.md
- bias-audit.md
- ethics-checklist.md

3.3.7 Packaging for Deployment

Steps

a. Export trained models to versioned directories.
b. Containerise each model with Docker.
c. Store model artefacts and datasets using DVC or MLflow model registry.

Deliverables:

- ml-model-registry/
- Dockerfiles for narrative, NER, and RE models

3.4 Phase 4: System Development

Goal: Build and integrate all core system components, back-end APIs, database layer, security, front-end UI, and visual analytics, and connect them to the ML inference services developed in Phase 3.

3.4.1 ML Service Integration (within System Development)

Goal: Operationalise trained models within the platform's back-end and link them to the FastAPI server.

3.4.1.1 API Wrapping

Steps

a. Build FastAPI microservices for each model:
   - /narratives → topic clusters
   - /entities → named entities
   - /relationships → entity links

b. Return JSON with fields: text, predictions, confidence, timestamp.

c. Add batch mode for historical data and real-time mode for streaming.

Deliverables:

- ml-inference-api/ repo
- api-spec.yaml (OpenAPI schema)

3.4.1.2 Integration with CoordinatorAgent

Steps

a. Update coordinator_agent logic to call model endpoints in sequence:
   - DataEngAgent → OsintAgent → LexiconAgent → TrendAnalysisAgent

b. Use mongodb_storage.py to store model outputs in analysis_results.

c. Log all inference results for audit and retraining triggers.

Deliverables:

- Updated ew_agents/agent.py
- integration-plan.md

3.4.2 Monitoring Hooks

Steps

a. Record:
   - Inference latency
   - Confidence scores
   - Model version used

b. Push metrics to GCP Monitoring / Prometheus.

Deliverables:

- ml-monitoring-hooks.py
- ml-metrics-dashboard/

3.4.3 Back-End Development

Steps

a. Set up FastAPI server structure
   - main.py as root entry point
   - Organise routers into modules: /analysis, /reports, /storage, /auth, /ml_inference

b. Develop REST API endpoints
   - Data access:
     GET /analyses, GET /analysis/{id}, GET /storage/stats
   - Analysis trigger:
     POST /run_analysis → calls CoordinatorAgent to orchestrate all ML tasks
   - Reports system:
     GET /api/reports/available, GET /api/reports/download/{analysis_id}

c. Implement business logic
   - Orchestrate DataEngAgent → OsintAgent → LexiconAgent → TrendAnalysisAgent
   - Store outputs in analysis_results collection on MongoDB

d. Optimise queries
   - Add MongoDB indexes (on timestamps, actors, narrative keywords)
   - Profile query latency and refactor heavy pipelines

Deliverables:
- api-spec.yaml (OpenAPI)
- ew_backend/ service directory
- mongodb_storage.py module with indexes and CRUD utilities
- performance-report.md (query profiling results)

3.4.4 Security & Privacy Compliance

Steps

a. Implement authentication & authorisation
   - OAuth2 + JWT for session tokens
   - Role-based access (admin, analyst, viewer)

b. Add data security controls
   - HTTPS/TLS everywhere
   - Encrypt MongoDB credentials using GCP Secret Manager
   - Principle of least privilege for IAM roles

c. Add audit logging
   - Log access attempts, login failures, API usage volumes

d. Document compliance alignment (GDPR, local data laws, organisational ethics)

Deliverables:
- auth-service/ module
- security-config.md
- audit-log-policy.md

3.4.5 ML Service Integration

Steps

a. Deploy trained models as containerised FastAPI inference services (from Phase 3).
b. Add an internal microservice route for CoordinatorAgent to call:
    - /narratives, /entities, /relationships
    - Include confidence scoring + fallback logic for low-confidence outputs.
c. Write integration middleware to:
    - Format outputs into the standard schema
    - Store results in MongoDB
    - Emit metrics to GCP Monitoring

Deliverables:

- ml-integration-middleware.py
- integration-plan.md
- Updated ew_agents/agent.py

3.4.6 Front-End Development (UI/UX)

Steps

a. Start from approved wireframes/mockups.
b. Build SPA (single-page app) using React or Vue:
    - Dashboard view
    - Search + filter panel
    - Narrative network graph
    - Timeline visualisation
c. Implement visual components:
    - Graphs (NetworkX/Cytoscape.js/D3.js)
    - Timelines (Plotly, Timeline.js)
    - Sentiment/narrative trend charts
d. Add filtering & search features
    - Keyword search
    - Filter by platform, actor, date
e. Ensure responsive design and accessibility (WCAG AA)
f. Connect to back-end APIs using Axios/Fetch with authentication tokens

Deliverables:

- frontend/ repo
- component-library/ (buttons, charts, cards)
- style-guide.md
- accessibility-checklist.md

3.4.7 Design System & UX Testing

Steps

a. Build a reusable design system / component library
    - Typography, colour palette, buttons, modals, charts
    - Document in Storybook or similar
b. Conduct usability tests with 5–10 analysts/journalists
c. Gather feedback on clarity, navigation, responsiveness
d. Iterate designs and update the style guide

Deliverables:

- design-system/
- usability-report.md
- Updated wireframes.pdf

3.4.8 Monitoring Hooks

Steps

a. Add system monitoring (Prometheus + Grafana or GCP Cloud Monitoring) for:
   - API latency
   - Error rates
   - Storage growth
   - ML inference metrics (confidence, model version)

b. Centralise logs (ELK stack or Cloud Logging)

c. Set alert thresholds and escalation protocols

Deliverables:

- monitoring-config/
- system-health-dashboard/

3.5 Phase 5: Testing & Quality Assurance (QA)

Goal: Build and integrate all core system components, back-end APIs, database layer, security, front-end UI, and visual analytics, and connect them to the ML inference services developed in Phase 3.

3.5.1 Test Planning

Steps:

a. QA Lead drafts a master test plan covering:
   - Functional testing (do features work?).
   - Integration testing (do APIs, front-end, back-end, and ML models connect properly?).
   - Regression testing (did new updates break old features?).
   - Performance testing (can the system handle expected load?).

b. Define test environments (staging vs production).

c. Create traceability matrix mapping features to test cases.

Checklist:

- Document all acceptance criteria.
- Assign testers to each module.
- Ensure test data is anonymised and ethically sourced.

3.5.2 Test Case Development

Steps**:**

a. Write detailed test cases for:
   - API endpoints (e.g., retrieval, filtering).
   - ML outputs (NER accuracy, narrative clustering).
   - Front-end UI flows (search, filter, graph navigation).
   - Auth/authz (role permissions, failed logins).

b. Prepare automated test scripts (e.g., Selenium for UI, Pytest for APIs, JMeter for load testing).

c. Store test cases in version control or test management tools (Jira, TestRail).

3.5.3 Test Execution & Bug Tracking

Steps:

a. Execute end-to-end test cycles.
b. Log bugs/issues in a central tracker (Jira, GitHub Issues, Linear).
c. Classify issues: *critical, major, minor*.
d. Retest after fixes; close tickets upon resolution.

3.5.4 Security & Compliance Testing

Steps:

a. Conduct penetration testing (manual + automated).
b. Run vulnerability scans (OWASP ZAP, Burp Suite, Nessus).
c. Validate data privacy rules: encryption, anonymisation, access controls.
d. Ensure compliance with organisational & legal standards.

3.5.5 Usability & UX Testing

Steps:

a. Conduct usability tests with UX team + target users.
b. Evaluate across devices (desktop, mobile, tablet).
c. Ensure consistent experience: navigation, font sizes, colours, accessibility features.
d. Gather structured feedback (surveys, SUS scores, heuristic evaluation).

3.5.6 ML Model Validation

Steps:

a. Monitor ML outputs during system tests:
   - Accuracy, precision, recall, F1.
   - Latency (response time).
   - Bias detection (fairness metrics).
b. Validate model integration into APIs.
c. Resolve inference/data issues (misclassification, broken links).

3.5.7 Automation & Continuous QA

Steps:

a. Automate critical test cases (CI/CD pipeline).
b. Run regression tests automatically on every code push.
c. Maintain QA dashboards to track coverage, defects, and performance.

3.5.8 Sign-Off & Release Readiness

Steps:

a. QA Lead compiles final QA report:
   - Passed/failed cases.
   - Open vs resolved issues.
   - Risk assessment.
b. Present findings to Project Manager + Leadership.
c. QA Lead gives final sign-off confirming release readiness.

Deliverables by end of Phase 5: 📌

- □ Master test plan + test case repository.
- □ Bug/issue tracker with resolved critical items.
- □ Security & compliance validation report.
- □ Usability findings + refinements logged.
- □ Automated test scripts integrated into CI/CD.
- □ QA Lead sign-off for deployment.

3.6 Phase 6: Deployment & Launch

This phase moves the system from staging into production, ensures infrastructure scalability, sets up monitoring, and coordinates communication with stakeholders.

3.6.1 Infrastructure Provisioning

Steps:

a. Select hosting environment (cloud provider, on-prem, or hybrid).
b. Provision production servers with scalability in mind (auto-scaling groups, container orchestration via Kubernetes/Docker).
c. Apply security best practices:
   - Firewalls, VPN access for admins.
   - Encryption (TLS/SSL).
   - Role-based access control.
d. Document infrastructure in an architecture diagram.

3.6.2 CI/CD Setup

Steps:

a. Configure Continuous Integration (CI) pipeline to:
   - Run automated tests on commits.
   - Block merges if critical tests fail.
b. Configure Continuous Deployment (CD) pipeline to:
   - Automatically deploy to staging.
   - Allow controlled/manual approval for production.
c. Ensure rollback strategy is in place (blue-green deployments, canary releases).

3.6.3 Back-End & ML Deployment

Steps:

a. Deploy back-end APIs to production servers.
b. Deploy ML models for inference:
   - Use containers or model serving frameworks (TensorFlow Serving, TorchServe, FastAPI).
   - Expose model endpoints via secured APIs.
c. Set up model monitoring pipeline:
   - Track latency, accuracy, and drift.
   - Log predictions for error analysis.
d. Configure retraining workflow: schedule periodic retraining or trigger on drift thresholds.

3.6.4 Front-End Deployment

Steps:

a. Deploy front-end applications to a hosting provider or CDN (e.g., Netlify, Vercel, Cloudflare, S3 + CloudFront).
b. Enable responsive checks for all major devices/browsers.
c. Integrate with back-end APIs and confirm data flows.
d. Implement caching/CDN optimisations for speed.

3.6.5 Monitoring & Logging

Steps:

a. Set up application monitoring (Prometheus, Grafana, Datadog, New Relic).
b. Configure centralised logging (ELK stack, CloudWatch, Loki).
c. Define alert thresholds for uptime, latency, error rates, and ML anomalies.
d. Establish on-call rotation/escalation protocol for incident response.

3.6.6 Documentation & Training

Steps:

a. Prepare user documentation:
   - Quick-start guide.
   - Feature walkthroughs.
   - FAQs/troubleshooting.

b. Develop technical documentation for maintainers:
   - API documentation.
   - Architecture diagrams.
   - Deployment scripts/workflows.

c. Conduct training sessions with key user groups (journalists, analysts, admins).

d. Provide support channels (email, Slack/Discord group, ticketing system).

3.6.7 Launch Communications

Steps:

a. Draft a launch announcement plan:
   - Internal stakeholders → email/slides.
   - External stakeholders → blog post, social media, press release if needed.

b. Coordinate with Project Manager for timing.

c. Provide demo sessions or live walkthroughs.

d. Gather immediate feedback post-launch.

3.6.8 Final Launch Sign-Off

Steps:

a. QA Lead verifies all pre-launch tests pass in production.

b. Tech Lead confirms infra & APIs are stable.

c. Ethics Lead signs off on compliance & governance.

d. Project Manager gives final go-live approval.

Deliverables by end of Phase 6 📌

□ Live production system (front-end, back-end, ML models).
□ CI/CD pipelines running with rollback strategy.
□ Monitoring & logging dashboards + alerting system.
□ Model monitoring & retraining workflow.
□ User + technical documentation.
□ Launch announcement & stakeholder comms.
□ Formal launch sign-off from leadership.

3.7 Phase 7: Post-Launch & Maintenance notes

Purpose: To give future developers/DevOps clear guidance on how to manage deployments, infrastructure, monitoring, and production operations for the ElectionWatch platform.

3.7.1 Deployment Operations

Environment Configuration

- Core environment variables required for production:
  - GOOGLE_CLOUD_PROJECT
  - GOOGLE_CLOUD_LOCATION
  - MONGODB_ATLAS_URI
  - *(Optional)* OPEN_ROUTER_API_KEY
- These are stored in:
  - .env file (for local dev- not committed to git)
  - GCP Secret Manager (for production-recommended)

Deployment Workflow

Standard steps to deploy a new release:# Build and push image
chmod +x deploy_gcloud.sh
./deploy_gcloud.sh

The script:

- Builds Docker image
- Pushes to Google Container Registry
- Deploys service to Cloud Run
- Configures memory/CPU, max/min instances
-

Release Checklist

Before merging to main:

- All tests pass (ml-test-suite, front-end, API)
- Dependencies are locked (requirements.lock)
- API docs updated (api-spec.yaml)
- Ethics & QA sign-off logged
-

After deploy:

- Validate /health endpoint
- Run smoke tests on all APIs
- Verify new image tag is live in Cloud Run console

3.7.2 ML Deployment Notes

Inference Services

Each model (narratives, NER, RE) is containerised and deployed as a Cloud Run microservice
Endpoints:

- /narratives
- /entities
- /relationships

CoordinatorAgent orchestrates calls to these during /run_analysi

Model Versioning

Store trained models in ml-model-registry/ with semantic versioning (v1.2.3)
Update model-card.md for each version
Record deployed version tags in deployment-log.md

Retraining

Trigger retraining when:

- Drift > threshold
- Accuracy < agreed baseline

Retraining steps:

- Fine-tune with new labelled data
- Rerun ml-test-suite
- Redeploy container with new tag

3.7.3 Monitoring & Logging

Monitoring Tools

GCP Cloud Monitoring dashboards:

- API latency
- Error rates
- Instance scaling
- ML inference confidence

Alerts configured for:

- 5% error rate
- Latency > 2s
- Model drift alerts

Logs

Centralised logs in Cloud Logging
Log retention: 90 days
Export weekly summaries to logs/weekly_reports/

3.7.4 Security & Access

Secrets & Credentials

Stored in GCP Secret Manager
Rotate credentials quarterly
Service accounts follow principle of least privilege

Access Control

- Role-based IAM:
  - dev → staging
  - ops → production
  - read-only → stakeholders
- No direct SSH/RDP access; use Cloud Console or gcloud CLI

3.7.5 Knowledge Transfer

handover_deploy.md should include:

- Cloud Run service names + URLs
- Secret Manager key names
- Contact list (dev, ops, on-call)
- Current model versions + drift status
- Deployment history (with tags and dates)

New DevOps Onboarding

- Review deploy_config.env.example
- Walk through deploy_gcloud.sh and rollback steps
- Read docs/API_GUIDE.md and docs/DEPLOYMENT.md
- Do a supervised test deployment to staging

Chapter 04 Handover & Maintenance Notes

Purpose: To document the key technical practices, extension points, and maintenance routines for the ElectionWatch system post-build.

4.1 System Extension Guidelines

4.1.1 Adding New API Endpoints

Add the route in main.py (FastAPI).
Place logic in a dedicated module under ew_backend/ (or ew_agents/ if agent-based).
Update api-spec.yaml (OpenAPI schema).
Write unit + integration tests before deployment.
Reflect the change in /docs/API_GUIDE.md.

4.1.2 Adding New Agents

Create a new _tools.py file in ew_agents/ (e.g. fact_check_tools.py).
Register the agent and its capabilities in ew_agents/agent.py.
Define inputs/outputs clearly to integrate with CoordinatorAgent.
Test locally using:adk run ml.main:coordinator_agent

4.1.3 Adding New ML Models

Follow Phase 3 workflow (training → evaluation → bias audit).
Wrap the model in a FastAPI microservice (like the others).
Register the new endpoint with the CoordinatorAgent orchestration flow.
Update monitoring hooks to log model version + latency.

4.1.4 Modifying the Front-End

Add new features as modular components in frontend/components/.
Maintain consistent styling by using tokens from design-system/.
Run npm run storybook (or equivalent) to preview UI changes.
Write Cypress/Selenium tests for new interactive features.
Conduct quick usability tests before merging to main.

4.1.5 Maintenance Best Practices

Dependencies

Update Python deps:pip install --upgrade -r requirements.txt
pip freeze > requirements.lock
Update front-end deps:npm update && npm audit fix

Database

Maintain MongoDB indexes (see mongodb_storage.py)
Archive old analysis_results after each election cycle
Run monthly storage health checks with /storage/stats

Monitoring & Logs

Review system dashboards weekly (API latency, errors, ML drift).
Rotate logs and audit access attempts quarterly.
Keep alert escalation contacts up to date.

Security
Rotate API keys, tokens, and service account credentials quarterly.
Validate all changes pass security and regression tests before production deploy.

Documentation
Update /docs folder whenever new features are added.
Keep README.md updated with system overview and setup steps.

Knowledge Transfer
Maintain a handover.md file listing:
- All active environment variables and secrets
- Locations of all deployed services
- Current maintainers and their responsibilities
- Recent known issues + backlog

Run a live walkthrough with new team members before handing over access.

4.1.6 Data Quality Monitoring

Steps:
a. Continuously monitor incoming data for:
   - Duplication.
   - Missing or corrupted entries.
   - Bias indicators.
b. Implement data quality dashboards.
c. Flag anomalies early and adjust acquisition pipelines as needed.

4.1.7 User Behaviour & Usage Analytics

Steps:
a. Track user behaviour with analytics (Google Analytics, Matomo, custom logging).
b. Analyse:
   - Most-used features.
   - Drop-off points in workflows.
   - Search/filter queries not yielding results.
c. Use findings to prioritise UI/UX improvements and feature rollouts.

4.1.8 Model Monitoring & Retraining

Steps:
a. Monitor ML model performance against baseline metrics.
b. Detect drift (input distribution changes, accuracy drop).
c. Schedule retraining cycles with updated labelled data.
d. Validate retrained models (bias, fairness, precision/recall).
e. Deploy new versions using CI/CD with rollback options.

4.1.9 Continuous System Maintenance

Steps:
a. Apply security patches regularly (servers, libraries, dependencies).
b. Update infrastructure (cloud services, container images).
c. Maintain API compatibility (versioning strategy).
d. Fix front-end/back-end bugs rapidly using triage (critical vs minor).
e. Run regression tests after every patch/update.

UI/UX Enhancements

Steps:

a. Conduct quarterly usability studies.
b. Roll out incremental improvements (navigation, visualisations, accessibility).
c. Update design system & documentation to reflect changes.
d. Validate enhancements with end-users before full rollout.

Narrative Curation & Accuracy Checks

Steps:

a. Periodically review extracted narratives for accuracy, nuance, and relevance.
b. Validate against ground-truth data or expert review panels.
c. Curate a narrative library/archive for long-term reference.
d. Update ML training datasets with curated examples to improve accuracy.

Deliverables by end of Phase 7

- □ Regular user feedback reports & feature roadmap.
- □ Monthly/quarterly system + impact performance reports.
- □ Continuous monitoring dashboards (data quality, model drift, usage).
- □ Updated, retrained ML models where required.
- □ Stable, patched system with minimal downtime.
- □ Improved UI/UX and updated design system.
- □ Curated narrative archive with accuracy validation.

Summary/conclusion:

ElectionWatch represents more than a technical intervention, it is a commitment to building safer, more transparent information ecosystems during elections. By merging AI tools, data science, and journalistic inquiry, the project strengthens the ability of media and civil society to identify, understand, and counter the evolving threats of electoral disinformation. In doing so, it bridges the gap between technology and accountability: transforming raw data into actionable insight, and chaos into clarity. The platform's design reflects Safety by Design principles prioritising ethical data use, privacy protection, and contextual sensitivity in every analytical step.

As ElectionWatch expands to cover additional geographies and languages, its value will lie not only in what it detects but also in what it enables: more informed reporting, earlier risk detection, and ultimately, a more resilient democratic process. The goal is simple but vital; to ensure that truth travels as fast as falsehood, and that those safeguarding elections are never outpaced by those seeking to undermine them.