CCExtractor v1.00 Release

GSoC 2026 Proposal

Table of Contents

1. About Me

  1.1 Personal Information

  1.2 Education

  1.3 Technical Skills

  1.4 Open Source & Experience

  1.5 Availability

2. Contributions to CCExtractor

3. Project Execution Strategy and Goals

4. Sample Platform Reliability Investigation & Fixes

  4.1 Current State

  4.2 Issue 1 — Empty Path False-Positive

  4.3 Issue 2 — Stale last_passed_on_windows Values

  4.4 Issue 3 — Baseline Desynchronization

  4.5 Issue 4 — Release Webhook Golden Update Is a Silent No-Op

  4.6 Summary and Fix Plan

5. C → Rust Redundancy Elimination

  5.1 Guiding Principle

  5.2 Tier 1 — Removable After Windows CI Migration

    5.2.1 Phase 0 — Windows CI Alignment (Prerequisite)

    5.2.2 Phase 1 — Deletion (After Phase 0)

  5.3 Tier 2 — Remove DISABLE_RUST Fallback Branches (~30 lines)

  5.4 Tier 3 — ES Wrapper Migration (Not Guaranteed Within Scope)

  5.5 Tier 4 — Conditional on Open PR Merges

  5.6 Out of Scope

5.7 Test Coverage Gaps Closed

6. Regression Test Coverage Expansion

6.1 Current State

6.2 Verified Coverage Gaps

6.3 Sample Acquisition Strategy

6.4 Incremental Integration Strategy

7. Flutter GUI Synchronization with Core

7.1 Current State

7.2 Key Synchronization Gaps

7.2.1 Gap 1 — Missing Input Format Support (SCC)

7.2.2 Gap 2 — Multi-page Teletext Support Mismatch (Critical)

7.2.3 Gap 3 — Missing Feature Exposure from Rust Implementation

7.2.4 Gap 4 — Unreachable GUI Feature (--analyzevideo)

7.3 Impact of Proposed Work

7.4 Synchronization Plan

7.5 Outcome

8. Automated Release Pipeline for CCExtractor 1.00

8.1 Current State: Extensive but Inconsistently Coordinated

8.2 Live Proof: What Users Actually Receive Today

8.3 Gap 1 — Version Triple Desync

8.4 Gap 2 — Build Provenance Lost in Release Binaries

8.5 Gap 3 — Self-Integrity SHA256 Broken in Installed Builds

8.6 Gap 4 — No macOS Release Artifact

8.7 Gap 5 — Trigger Mismatch Creates Missing and Incomplete Releases

8.8 Gap 6 — AButler/upload-release-assets@v3.0 Is Unmaintained

8.9 Gap 7 — No SHA256 Checksums on Release Assets

8.10 Gap 8 — Docker Images Built But Not Distributed

8.11 Summary

9. Project Timeline and Deliverables

---

1. About Me

1.1 Personal Information

Name	Chandragupt Singh
Location	Ujjain, Madhya Pradesh, India
Email	chandraguptsingh1729@gmail.com
GitHub	github.com/x15sr71
LinkedIn	linkedin.com/in/chandraguptsingh

1.2 Education

I am currently a third-year undergraduate student pursuing a B.Tech in Computer Science and Engineering.

1.3 Technical Skills

Languages: C, Rust, Python, TypeScript, Dart

Build & CI: CMake, GitHub Actions, Docker

Cloud & Platforms: Google Cloud Platform (GCP)

Core Interests: Backend Development, Systems Programming, Distributed Systems, and Developer Infrastructure

1.4 Open Source & Experience

I discovered CCExtractor in October 2025 and have been actively contributing since then across different parts of the project, primarily focusing on the core extraction tool.

As part of my GSoC preparation, I have been actively auditing the codebase and testing infrastructure, identifying issues in the testing pipeline and working on their fixes. This work directly informs the execution plan in this proposal.

I have also built full-stack production-grade projects:

SyncIt — A cross-platform music synchronization system. (Backend, Client) - Designed a modular Express.js backend supporting playlist syncing, OAuth authentication, and scalable architecture with planned Redis caching. Developed a Next.js frontend for managing and

visualizing cross-platform music data with a modern UI.
Link – Backend , Frontend

I enjoy working on real-world engineering problems involving system design, scalability, and reliability.

WildGuard — A platform for connecting people with NGOs and volunteers for wildlife rescue. (Backend, Client). Implemented a React + Vite frontend with Google Maps integration and AI-based animal identification. Built a scalable backend using Node.js, PostgreSQL, and Prisma, integrating Google Gemini and Cloud Vision APIs for image recognition and analysis.

Link – Backend , Frontend

1.5 Availability

I am available to dedicate ~30–40 hours per week during the GSoC coding period and can adjust my schedule as required.

---

2. Contributions to CCExtractor

Packaging

- PR #260293 — Re-introduced CCExtractor to Homebrew after a two-year absence. Coordinated upstream-system-libs support across macOS and Linux, resolved utf8proc include path failures flagged during review, and passed full brew audit --strict, brew test, and brew linkage verification on both Apple Silicon and Linux.
- PR #1998 — Introduced Snapcraft-based packaging with a GitHub Actions workflow, enabling reproducible, cross-distribution Linux builds with bundled runtime dependencies and deterministic library resolution.
- PR #1985 — Added Homebrew installation instructions to COMPILATION.MD.

Features

- PR #2020 — Added structured JSON output for -out=report, enabling machine-readable caption metadata for automation workflows with a versioned schema and full multi-program stream support.
- PR #1773 — Restored XMLTV generation for ATSC EIT/VCT streams and corrected EIT bounds checks.
- PR #1837 — Added ATSC VCT virtual channel numbers and call signs to XMLTV output.
- PR #1885 — Implemented guarded ASS/SSA \pos positioning tags for CEA-608 caption placement.
- PR #2100 — Corrected BCD parsing for DVB EIT start times in XMLTV output.

Sample Platform Reliability

- PR #1074 — Identified and traced a silent false-positive chain through Processor.cs, ServerComparer.cs, and Tester.cs — initiated the fix by defining tempFolder in variables.bat, with follow-up infrastructure fixes in progress.

Build Fixes

- PR #1756 — Fixed ARM64/AArch64 build failure caused by platform-dependent c_char signedness (i8 on x86_64, u8 on ARM64). Changed c_char::from(0i8) to 0 as c_char in avc/nal.rs, verified on both architectures.
- PR #1907 — Added -system-libs flag for Linux builds, enabling CCExtractor to link against system-installed libraries for package manager compatibility.
- PR #1922 — Fixed utf8proc header path resolution under -system-libs builds, unblocking the Homebrew reintroduction.

Stability & Error Handling

- PR #2134 Fixed SPUPNG output correctness by aligning file indexing with spec (sub0000), preventing counter advancement on empty EOD frames, and normalizing header paths for deterministic CI output.
- PR #1769 — Added HEVC/H.265 stream type recognition to prevent unhandled stream type crashes on ATSC 3.0 streams.
- PR #2138 — Added bounds checking in timing.rs and c_functions.rs to prevent Rust panics on out-of-range timestamps.
- PR #2151 — Implemented lazy CEA-708 service decoder allocation to prevent OOM panics on x86 Windows.

Flutter

- PR #12 — Resolved macOS build failures blocking development on modern systems: updated deployment target from 10.12 to 10.15, integrated CocoaPods for Firebase SDK linking, and modernized AppDelegate.swift for Xcode 16 + Apple Silicon compatibility.
- PR #853 (OPEN) —Modernized the Android build system (Gradle, AGP, Kotlin) and upgraded the full Flutter dependency stack, while refactoring Google Sign-In and fixing compatibility issues to restore successful builds on current toolchains.

---

3. Project Execution Strategy and Goals

The goal of this project is to make CCExtractor's development pipeline reliable so future changes can be made with confidence.

The work is structured in stages, each building on the previous:

- **Fix CI Reliability:** Resolve false positives, incorrect PR test reporting, and mismatches between UI and GitHub status. This establishes a trustworthy testing system.
- **Validate and Update Test Results:** Identify real regressions vs genuine improvements. Fix real code regresssions in CCExtractor and Update expected outputs where current results are more accurate.
- **Remove Redundant Code Paths:** Clean up legacy and unused code (especially in C → Rust transition areas) to improve maintainability.
- **Expand Test Coverage:** Cover untested paths using additional samples where needed.
- **Synchronize Flutter GUI with CLI:** Ensure feature parity and consistent behavior across interfaces.
- **Improve Release Pipeline:** Streamline build and release workflows across platforms.

Each stage will be validated through passing CI runs before moving forward, ensuring all changes are built on a stable foundation.

The analysis in this proposal is based on sample-platform testing, logs, and cross-repository inspection. While infrastructure-specific details may vary slightly in production, the approach is designed to adapt while maintaining correctness at every step.

---

4. Sample Platform Reliability Investigation & Fixes

4.1 Current State

The sample platform's CI bot — which comments "this PR breaks X" or "this PR fixes Y" on every pull request — is the primary tool mentors and contributors use to assess whether a change is safe to merge. If that signal is untrustworthy, every review decision carries hidden risk: regressions get merged, legitimate fixes get blocked, and the test suite's coverage numbers overstate real confidence.

Before writing this proposal, the full end-to-end test workflow — from PR trigger through VM execution, result upload, and bot comment generation — was traced, and five compounding issues that explain why the bot's attribution has been unreliable were identified. The investigation was posted to the Zulip channel for validation; Maintainer Willem confirmed the root cause analysis was correct and the first fix PR (PR #1074) was subsequently merged.

Key finding: the bot's attribution logic is correct. The problem is that the three inputs it relies on — last_passed_on_*, the master test baseline, and the golden/expected output files — silently drift out of sync due to independent bugs across ccx_testsuite and the platform.

Evidence of the problem in practice: In the recently merged PR #2210 (stdin null pointer fix), the bot reported multiple unrelated regressions in file-based tests (--autoprogram --out=ttxt, --out=spupng, --startcreditstext), despite the change being limited entirely to stdin handling. The same --startcreditstext test was classified as "PR breaks this" on Linux and "merging would fix this" on Windows — the same test, the same PR, opposite conclusions. That contradiction is a direct symptom of the baseline divergence described below.

4.2 Issue 1 — Empty Path False-Positive

Root Cause: variables.bat did not define tempFolder. When %tempFolder% expanded to an empty string, ccx_testsuite received an invalid path, causing Processor.cs to record missing outputs as empty strings. This fed into ServerComparer.cs throwing ArgumentException, which Tester.cs caught and swallowed — silently skipping the file comparison upload for that specific output. However, the test still sends exit code and runtime via SendExitCodeAndRuntime, resulting in a partial upload without TestResultFile entries.

Additionally, the platform's status computation counted only non-null TestResultFile rows, making missing rows indistinguishable from zero failures.

Additionally, runCI.bat does not ensure that %tempFolder% exists before execution. If the directory is missing, output files are never written even with a correct path. Adding an explicit mkdir step ensures deterministic behavior.

Evidence: SPUPNG fails consistently on Linux master runs but passes on Windows master runs with the same golden files and the same commit. Examination of both Linux and Windows CI logs confirms the pattern: "Empty path name is not legal" appears on both platforms for the same affected tests, followed by exit code 0. The underlying empty path crash is cross-platform; however, the platform's status computation previously failed to account for missing results, leading to false positives in GitHub status reporting.

Progress: PR #1074 (merged) defines tempFolder in variables.bat. Follow-up PR #1081 (closed) proposed CI deployment; Maintainer Willem manually updated variables.bat on the platform instead. A follow-up PR to unify status computation using get_test_results() as the single source of truth ensures the CI system correctly reports failures even when output files are missing, regardless of underlying cause.

This ensures that even if upstream failures occur (e.g., decoder differences or filename mismatches), the CI system will never report false positives and remains a reliable signal for contributors and maintainers.

4.3 Issue 2 — Stale last_passed_on_windows Values

Root Cause: When Windows tests fail to upload comparison results (because Issue 1 causes comparisons to be skipped), get_test_results() correctly detects the absence of TestResultFile entries and marks those tests as errors. As a result, the affected tests are excluded from test_ids_to_update in update_build_badge():

Tests that never upload results never advance last_passed_on_windows. Over time this value becomes stale or remains unset — and since last_passed_on_windows is a direct input to PR attribution, stale values cause the bot to misclassify whether a failure is pre-existing or newly introduced by the PR.

Fix: Once missing-output scenarios are correctly handled and comparisons execute as expected, last_passed_on_windows and last_passed_on_linux advance automatically on successful comparison runs — no additional code change required, as both platforms are affected equally.

4.4 Issue 3 — Baseline Desynchronization

Root Cause A — Race Condition: get_info_for_pr_comment() queries the latest completed master test at comment-generation time — a live query, not a snapshot taken at PR test creation:

If a new master commit test completes while the PR test is still running, last_test_master shifts. A test that was in common_failed_tests (pre-existing failure — PR not at fault) can flip into extra_failed_tests (PR broke this) purely because the baseline moved.

Root Cause B — Baseline Mismatch: Two different baselines are in play simultaneously:

Test execution (VM side): GeneralData['fetch_commit_{platform}'] → last tested master commit → TestResultFile.got (or stored golden as fallback)

PR comment classification: last_test_master (live query) + last_passed_on_{platform} on RegressionTest

These are computed differently and updated at different times. If a master commit test partially fails, fetch_commit_* advances but last_passed_on_* does not update for the failed tests.

Fix: Snapshot last_test_master.id when the PR test row is created and store it on the Test record. get_info_for_pr_comment() reads the stored value instead of re-querying live. This single change resolves both the race condition and baseline mismatch.

4.5 Issue 4 — Release Webhook Golden Update Is a Silent No-Op

Root Cause: Every release is supposed to update RegressionTestOutput.correct (the stored golden file hash used as a fallback baseline when no recent master result exists) to reflect what the new release actually produces. The release webhook handler in mod_ci/controllers.py contains two independent bugs that cause this to silently do nothing:

Bug A — Python Identity Check Instead of SQL Filter:

test_result_file.c.got is not None is a Python identity check on a SQLAlchemy column object. The column object itself is never None, so this always evaluates to True in Python and no SQL WHERE clause is generated. The correct form is .isnot(None).

Bug B — .values() Is a SELECT, Not an UPDATE:

.values() on a SQLAlchemy Query object is a SELECT operation — it returns rows as tuples. It does not write anything to the database. The correct call is .update({RegressionTestOutput.correct: test_result_file.c.got}).

safe_db_commit() runs afterward and logs "Successfully updated baseline tests for release!" — but nothing was dirty to commit, so nothing happened.

Practical Consequence: Golden output hashes do not advance with releases. If CCExtractor's output changes between releases (which it does, intentionally), golden files silently diverge. Tests fail not because anything is broken in the PR under review, but because the expected files never updated.

Fix: Change the query target from RegressionTestOutput.correct (a column) to RegressionTestOutput (the model), replace .values() with .update(), and replace is not None with .isnot(None) for both the RegressionTestOutput.correct and RegressionTest.expected_rc updates in the webhook handler.

4.6 Summary and Fix Plan

Issue	Root Cause	Fix	Status
1 — Windows false positives	tempFolder undefined + directory validation	Define tempFolder; mkdir before tester invocation	PR #1074 merged; manual update live; server-side fix PR in progress
2 — Stale last_passed_on_windows	Depends on Issue 1	Resolved by Issue 1 directory fix	Blocked on directory resolution
3 — Baseline desync	Live DB query + VM/attribution mismatch	Snapshot baseline ID at PR test creation	PR to be raised
4 — Golden update no-op	Two SQLAlchemy bugs in release webhook	Fix query target, .update(), .isnot(None)	PR to be raised

---

5. C → Rust Redundancy Elimination

5.1 Guiding Principle

A C module is removal-eligible only if: a Rust equivalent is active in the default build, the execution path is proven from main(), behavior is regression-tested, and no active FFI dependency remains. No new Rust rewrites are proposed.

5.2 Tier 1 — Removable After Windows CI Migration

Target: ccx_dtvcc.c (157 lines) + ccx_decoders_708.c (1852 lines)

Legacy C implementations remain compiled but Rust paths are active by default (DISABLE_RUST not set in CMake builds), making these functions effectively unused in typical execution — but build_windows.yml:124 defines DISABLE_RUST via /p:ExtraDefines=DISABLE_RUST (MSBuild, invisible to CMake analysis), making the C decoder active in Windows release artifacts. ccx_decoders_708_output.c is not a target — dtvcc_writer_init() is live at ccx_encoders_common.c:699.

Condition	Status
No Rust extern "C" block references these three specific dead functions (dtvcc_init, dtvcc_process_data, dtvcc_free)	Yes — build.rs allowlist uses a broad DTVCC regex (.*_?[Dd][Tt][Vv][Cc][Cc]_.*) that includes these symbols; the C code instead delegates to ccxr_dtvcc_init, making legacy function names unreachable at runtime
Dead in Linux/macOS default builds	Yes — DISABLE_RUST — 0 matches in all CMakeLists.txt
Active in Windows CI	Warning — build_windows.yml: 124 — C decoder live on Windows today
Not triggered by Linux/macOS regression suite	Yes — CEA-708 tests using --service and --708 --out=srt route through Rust path
Rust call chain proven from main()	Yes — ccx_decoders_common.c:303 → ccxr_dtvcc_init exported in src/rust/src/lib.rs → instantiates DtvccRust in decoder/mod.rs
Rust path tested	Yes — 8 unit tests at decoder/mod.rs:749–860 covering legacy API and DtvccRust (state persistence and lazy allocation)

5.2.1 Phase 0 — Windows CI Alignment (Prerequisite)

- Remove /p:ExtraDefines=DISABLE_RUST from build_windows.yml:124
- Confirm CEA-708 tests pass through DtvccRust on Windows CI after DISABLE_RUST removal
- Mark DISABLE_RUST fallback pattern deprecated in docs/Rust_migration_guide.md — Phase 0 completes the migration the guide describes; the #else branches are incomplete migration artifacts that this work finalizes

5.2.2 Phase 1 — Deletion (After Phase 0)

- lib_ccx/CMakeLists.txt:60 — replace aux_source_directory(...) with explicit set(SOURCEFILE ...) excluding both files
- Remove dead #else blocks: ccx_decoders_common.c:230–231, 305–311 · mp4.c:754–756
- Delete both files · run full 241-test suite on Linux, macOS, Windows

No new Rust code required. The full 241-test suite provides high-confidence validation across mixed-path scenarios (MP4, timing, formatting, encoder interactions) in addition to dedicated CEA-708 tests.

5.3 Tier 2 — Remove DISABLE_RUST Fallback Branches (~30 lines)

Depends on Tier 1 Phase 0. Once DISABLE_RUST is gone from all supported configurations, every #ifndef DISABLE_RUST / #else / #endif block is unconditionally dead.

Locations: ccx_decoders_common.c:227–232, 302–311, 491–512 · general_loop.c:646–680 (5 locations) · networking.c:49–63 (10+ blocks)

Note: ccx_decoders_common.c contains full dual-branch blocks (#ifndef/#else/#endif); general_loop.c and networking.c use guard-only blocks (#ifndef/#endif) without a C fallback in those regions.

5.4 Tier 3 — ES Wrapper Migration (Not Guaranteed Within Scope)

Target: es_functions.c wrapper bodies — every function is a one-line delegate:

Rust is the runtime implementation but general_loop.c calls C entry points directly — call-site migration required before removal.

Deliverables:

- 7 #[test] functions in src/rust/src/es/ covering ccxr_search_start_code, ccxr_next_start_code, ccxr_read_seq_info, ccxr_read_gop_info, ccxr_read_pic_info, ccxr_read_eau_info, ccxr_read_pic_data
- New regression test --in=ts --out=null --parsedebug — T209 (--parsedebug --out=srt) exists but mixes parse + encode paths; --out=null isolates parse correctness
- Incremental general_loop.c call-site migration — one function at a time, full regression run after each; removal only if all tests pass

5.5 Tier 4 — Conditional on Open PR Merges

Targets: ccx_encoders_srt.c, ccx_encoders_xds.c — not removable until Rust replacements are wired into ccx_decoders_common.c:373–399 dispatch.

- PR #2227 — SRT Rust port · Review required
- PR #1927 — SRT Rust port · Changes requested
- PR #2088 — XDS Rust module · Review required

Deliverable: Review both SRT PRs, select stronger implementation, wire into dispatch, verify ~80 --out=srt tests (T1–T136). Deletion only on zero new failures.

5.6 Out of Scope

C File	Lines	Reason
ccx_decoders_608.c	1359	No Rust equivalent; do_cb() called from Rust via FFI
ccx_decoders_708_output.c	753	dtvcc_writer_init() live at ccx_encoders_common.c:699
telxcc.c	1737	Rust has type stubs only
dvb_subtitle_decoder.c	2067	No Rust equivalent
mp4.c	1295	No Rust equivalent
matroska.c	2088	No Rust equivalent
general_loop.c	1795	No Rust equivalent; C entry point for all decode paths

5.7 Test Coverage Gaps Closed

Gap	Current State	Delivered By
Rust ES functions have no unit tests	Indirect via C regression only	Tier 3: 7 #[test] in src/rust/src/es/
No test isolates ES parsing from encoder output	T209 mixes both paths	Tier 3: --in=ts --out=null --parsedebug
Windows CI not validated against Rust decoder	build_windows.yml:124 uses DISABLE_RUST	Tier 1 Phase 0
DISABLE_RUST pattern documented as active	docs/Rust_migration_guide.md:52	Tier 1 Phase 0

CEA-708 Rust path coverage — tests using --service and --708 --out=srt route through ccxr_process_cc_data; 8 unit tests at decoder/mod.rs:749–860 covering legacy API and DtvccRust

---

6. Regression Test Coverage Expansion

6.1 Current State

CCExtractor's regression suite (T1–T241) provides strong coverage for Transport Stream (TS)-based workflows and common output formats (SRT, TTXT, SAMI). However, analysis of the InFormat and OutFormat enums in args.rs, the argument parser in parser.rs, and the full Rust module tree reveals that this coverage is concentrated almost entirely within one execution pipeline: TS input → caption decoder → text output.

Large portions of the codebase — including multiple demuxers, output encoders, and Rust-critical modules — remain completely unexercised in CI.

6.2 Verified Coverage Gaps

Input Format Gaps:

Code Path	Command	Sample Type	Sample Source
MKV demuxer (matroska.c)	ccextractor --in=mkv sample.mkv	MKV with embedded subtitles	FFmpeg samples
MXF demuxer (ccx_demuxer_mxf.c, 679 lines)	ccextractor --in=mxf sample.mxf	Broadcast MXF with captions	FFmpeg samples
SCC input (demuxer/scc.rs)	ccextractor --in=scc sample.scc	SCC caption file	Generated / sample repo

One additional path is invoked via auto-detection, not a CLI flag: GXF demuxer (ccx_gxf.c, 1413 lines) triggered by ccextractor sample.gxf using GXF broadcast file samples from FFmpeg samples.

Note: VobSub decoding (vobsub_decoder.c, 517 lines) is triggered internally within MKV processing and will be exercised by selecting MKV samples that contain VobSub subtitle tracks.

Implementation will prioritize widely used and easily reproducible formats (MKV, SCC, and output encoders) before extending to less common or harder-to-source formats such as GXF.

Output Format Gaps:

Code Path	Command	Sample Needed
SSA encoder (ccx_encoders_ssa.c, 383 lines)	ccextractor input.ts --out=ssa	Any existing TS sample from T1–T241
WebVTT encoder (ccx_encoders_webvtt.c, 572 lines)	ccextractor input.ts --out=webvtt	Any existing TS sample from T1–T241
CCD output (write_cc_buffer_as_ccd in ccx_encoders_scc.c)	ccextractor input.ts --out=ccd	TS with 608 captions from T1–T241

These three require no new samples — existing TS inputs already in the suite can be reused with a different --out= flag.

Feature Path Gaps:

Flag	Command	Sample Needed	Evidence
--list-tracks	ccextractor --list-tracks input.ts	Multi-track TS	track_lister.rs:834, args.rs:376
--splitbysentence	ccextractor input.ts --splitbysentence	Dialogue-heavy TS	args.rs:529

Rust-Specific Gap:

The ES parsing functions — read_seq_info (es/seq.rs:177), read_gop_info (es/gop.rs:24), read_pic_info (es/pic.rs:149) — are the actual runtime implementation that C delegates to via FFI exports at es/mod.rs:37–67. These files contain zero #[cfg(test)] blocks. Their correctness is never independently verified — they are only indirectly exercised through end-to-end TS regression tests that validate final output, not the parsing logic in isolation.

Dedicated Rust unit tests for these functions are delivered as part of the ES Wrapper Migration work described in Tier 3 of the C → Rust Redundancy Elimination section.

6.3 Sample Acquisition Strategy

Sample	Source	Used For
MKV with subtitles	samples.ffmpeg.org/Matroska/	--in=mkv test
MXF with captions	samples.ffmpeg.org/MXF/	--in=mxf test
GXF broadcast file	samples.ffmpeg.org/GXF/	GXF auto-detection path
SCC caption file	Generatable locally	--in=scc test
Existing TS samples	Already in suite (T1–T241)	--out=ssa, --out=webvtt, --out=ccd, --list-tracks, --splitbysentence

All new samples will be kept minimal in size for CI efficiency and will produce deterministic, reproducible outputs.

6.4 Incremental Integration Strategy

Introducing tests for previously untested code paths will surface latent bugs. A staged approach prevents CI disruption:

- Phase 1 — Non-blocking: New tests added as informational, non-blocking entries to expose discrepancies without breaking the pipeline.
- Phase 2 — Stabilization: Failures categorized as implementation bugs, sample issues, or expected edge-case behavior; issues fixed or expected outputs documented.
- Phase 3 — Enforcement: Stabilized tests converted to strict pass/fail regression checks and integrated into CI.

---

7. Flutter GUI Synchronization with Core

7.1 Current State

CCExtractor consists of a core CLI-based engine (C + Rust) and a Flutter-based GUI. The GUI interacts with the core binary entirely through CLI arguments, making synchronization between the two essential for correctness and feature completeness.

A joint analysis of the core (args.rs, parser.rs) and GUI (settings_model.dart, constants.dart, settings_repository.dart, input_settings.dart) codebases — including grep searches across

all of lib/screens/ — reveals that while the core CLI continues to evolve through Rust-based implementations, several features and capabilities are not fully reflected in the GUI.

7.2 Key Synchronization Gaps

7.2.1 Gap 1 — Missing Input Format Support (SCC)

The core defines InFormat::Scc in the InFormat enum (args.rs:1009–1010) and accepts it via the --in / --input CLI argument (args.rs:235–236, #[derive(ValueEnum)] on InFormat).

The GUI input format list in lib/utils/constants.dart:105–118 contains ['auto/default', 'ts', 'ps', 'es', 'asf', 'wtv', 'bin', 'raw', 'mp4', 'm2ts', 'mkv', 'mxf'] — 'scc' is absent. A grep across ccextractorfluttergui/lib/ for "scc" in input format context confirms zero results.

Impact: Users cannot explicitly select SCC input processing from the GUI despite full core support.

7.2.2 Gap 2 — Multi-page Teletext Support Mismatch (Critical)

The core defines tpage as pub tpage: Option<Vec<u16>> with action = clap::ArgAction::Append (args.rs:861–862), allowing multiple independent --tpage arguments (e.g., --tpage 891 --tpage 892). parser.rs:1204–1223 iterates the resulting vector and pushes each entry to tlt_config.user_pages.

The GUI models tpage as String tpage (settings_model.dart:111) with a single mapping 'tpage': '--tpage' (settings_model.dart:344). The GUI can only ever emit one --tpage <value>.

Impact: Multi-page simultaneous teletext extraction is entirely inaccessible from the GUI. Single-page usage (--tpage 891) remains backward compatible.

7.2.3 Gap 3 — Missing Feature Exposure from Rust Implementation

Five CLI flags defined in args.rs and parsed in parser.rs have no corresponding field, mapping, or UI control anywhere in the Flutter GUI codebase (confirmed by grep across ccextractorfluttergui/lib/):

Flag	Core Reference	Parser Reference
--scc-framerate	args.rs:302–303	parser.rs:846–861
--scc-accurate-timing	args.rs:309–310	parser.rs:864–866
--ocr-line-split	args.rs:653–654	parser.rs:752–753
--no-ocr-blacklist	args.rs:659–660	parser.rs:756–758
--tpages-all	args.rs:866–867	parser.rs:1225–1227

All five are Rust-only definitions — not present in any legacy C flag set.

Impact: Advanced SCC output compliance (--scc-accurate-timing), OCR accuracy controls, and bulk teletext extraction remain inaccessible to GUI users. Feature parity degrades as Rust-based capabilities expand.

7.2.4 Gap 4 — Unreachable GUI Feature (--analyzevideo)

The GUI defines bool analyzevideo in settings_model.dart:40, with a constructor default of false (settings_model.dart:175), correct CLI mapping 'analyzevideo': '--analyzevideo' (settings_model.dart:290), and full plumbing through copyWith, toJson, and fromJson (lines 429, 542, 661, 777).

A grep for "analyzevideo" across ccextractorfluttergui/lib/screens/, lib/bloc/, and lib/utils/ returns zero results. No widget in any settings screen references this field. The flag will always be false and --analyzevideo will never be emitted by the GUI.

Impact: The feature is fully modeled but permanently unreachable from the interface.

7.3 Impact of Proposed Work

Change Area	GUI Impact
CI reliability fixes	None — infrastructure changes only
Regression coverage expansion	None — adds test data and expected outputs only
C → Rust migration	Introduces new CLI features (table above) not reflected in GUI
Release pipeline fixes	None — build and distribution level only

The primary source of GUI desynchronization is the ongoing Rust migration, which introduces new CLI capabilities in args.rs that must be surfaced in the GUI to maintain parity.

7.4 Synchronization Plan

Priority 1 — Critical

- Update --tpage handling to support multiple values:
- settings_model.dart — change String tpage to List<String> tpage
- settings_repository.dart — emit multiple --tpage args in getParamsList()
- input_settings.dart — update text field to accept comma-separated pages

Priority 2 — Feature Parity

- Add 'scc' to inputFormats in lib/utils/constants.dart:105–118
- Add sccFramerate, sccAccurateTiming fields to settings_model.dart with corresponding UI controls in input_settings.dart and dropdown values in constants.dart
- Add ocrLineSplit, noOcrBlacklist, tpagesAll bool fields to settings_model.dart with toggle controls in hardsubx_settings.dart / input_settings.dart
- Surface the existing analyzevideo field in input_settings.dart via a CustomSwitchListTile

7.5 Outcome

Completing this sync ensures all core capabilities introduced during the GSoC period are accessible via the GUI, the Rust migration does not create hidden functionality gaps, and the GUI remains a reliable interface for all supported workflows — which is a prerequisite for a production-ready CCExtractor 1.00 release.

---

8. Automated Release Pipeline for CCExtractor 1.00

8.1 Current State: Extensive but Inconsistently Coordinated

CCExtractor already has a mature, multi-platform release infrastructure built across fifteen GitHub Actions workflows. On every release, the following artifacts are automatically produced and distributed:

- Windows: release.yml ('Upload releases') builds x64 and x86 MSI installers (WiX v4) and portable ZIPs with Flutter GUI bundled
- Linux packaging: build_appimage.yml, build_deb.yml (Ubuntu 24.04), build_deb_debian13.yml (Debian 13 Trixie), build_linux_systemlibs.yml, and build_snap.yml all trigger on release: published
- Package managers: homebrew.yml bumps the Homebrew formula, publish_chocolatey.yml pushes to Chocolatey (chained via workflow_run), publish_winget.yml submits a winget manifest

This is a real, working system that already ships to thousands of users. The goal for 1.00 is not to replace it — it is to fix the coordination gaps that currently cause incorrect artifacts to ship and prevent the pipeline from being trustworthy for a major release.

8.2 Live Proof: What Users Actually Receive Today

Before writing a single line of proposal, the official release was installed on a development machine (Apple Silicon, macOS Tahoe):

$ brew install ccextractor

Pouring ccextractor--0.96.6.arm64_tahoe.bottle.tar.gz 0.96.6

$ ccextractor --version

Version: 0.96.5

0.96.5

Git commit: Unknown

File SHA256: e3b0c44298fc1c149afbf4c8996fb924...

bytes

The official Homebrew bottle is tagged 0.96.6. Every macOS user who ran brew install ccextractor after the v0.96.6 release (February 19, 2026) received a binary that misreports

its own version. This is reproducible by any reviewer with a Mac and traces directly to specific lines in the current pipeline.

8.3 Gap 1 — Version Triple Desync

Severity: Critical

Root Cause: Version is defined independently in three source locations and no process enforces they are synchronized before a release tag is cut:

release.yml correctly reads the version from the git tag, so the Windows MSI filename embeds 0.96.6. But lib_ccx.h — which drives what the binary prints at runtime — was never updated. The desync is architecturally embedded.

Why a post-tag prepare job cannot fix this: The release: published trigger fires after the git tag already exists. A workflow that modifies lib_ccx.h and pushes creates a new commit the tag does not point to. Source tarballs downloaded at the tag still contain the stale version.

Proposed Fix — Pre-tagging version sync PR:

- Contributor opens PR: 'release: prepare v1.00' updating lib_ccx.h, CMakeLists.txt, snapcraft.yaml, and CHANGES.TXT
- PR is reviewed and merged to master
- A CI gate asserts the VERSION string in lib_ccx.h matches the intended tag — blocks release if files are stale
- Developer cuts tag v1.00 on the merge commit
- release: [published] fires — all builds run against the correct commit where every version source is synchronized

8.4 Gap 2 — Build Provenance Lost in Release Binaries

Severity: High

Evidence: The Homebrew bottle reports Git commit: Unknown. Homebrew builds from a .tar.gz source tarball with no .git/ directory — any git rev-parse call returns empty. This affects every release binary distributed through Homebrew.

Proposed Fix — Explicit commit injection in the GitHub Actions build step:

This works because GitHub Actions builds run against a full git checkout. Package manager builds from tarballs without .git/ correctly show Unknown — consistent with how other major open source projects handle this.

8.5 Gap 3 — Self-Integrity SHA256 Broken in Installed Builds

Severity: Medium

Evidence: File SHA256: e3b0c44298fc1c149afbf4c8996fb924... is the SHA256 of zero bytes. The binary fails to correctly resolve its own executable path under Homebrew's symlinked

install layout, consistent with incorrect path resolution relying on argv[0] without resolving symlinks.

Proposed Fix: Replace binary path resolution with realpath(argv[0], NULL) on POSIX systems and GetModuleFileName(NULL, ...) on Windows before hashing, ensuring the correct absolute path regardless of invocation method or install layout.

8.6 Gap 4 — No macOS Release Artifact

Severity: High

build_mac.yml has no release: event trigger. Despite having eight CI jobs that run on every push, macOS produces zero artifacts on GitHub Releases. Every other platform has a release artifact — macOS is the sole exception, and macOS users must build from source.

The fix targets the build_shell job — it uses the primary ./build.command script and already stages the binary to ./mac/artifacts/. It needs one trigger addition and one upload step, matching the exact pattern used by build_deb.yml and build_snap.yml.

8.7 Gap 5 — Trigger Mismatch Creates Missing and Incomplete Releases

Severity: Medium

Exact triggers from actual workflow files:

Workflow	Trigger
release.yml	release: types: [created]
build_snap.yml	release: types: [published]
build_appimage.yml	release: types: [published]
build_deb.yml	release: types: [published]
build_deb_debian13.yml	release: types: [published]
build_linux_systemlibs.yml	release: types: [published]
homebrew.yml	release: types: [published]

- Direct publish: created fires first, then published — release.yml starts building before other workflows, causing a temporary state where only Windows artifacts are available.
- Draft → Publish (common maintainer workflow): created is not fired at all — release.yml is skipped entirely, resulting in a release with Linux artifacts but no Windows MSI.
- Hidden dependency: publish_chocolatey.yml and publish_winget.yml trigger via workflow_run: workflows: ["Upload releases"]. Any rename of release.yml's name: field silently breaks the entire Chocolatey and winget distribution chain.

8.8 Gap 6 — AButler/upload-release-assets@v3.0 Is Unmaintained

Severity: Medium

release.yml uses AButler/upload-release-assets@v3.0 for all release uploads. Every workflow added since has migrated to softprops/action-gh-release@v2. Part of the

consolidation work is migrating the Windows and source tarball uploads to match, giving the entire release pipeline a single, maintained upload mechanism.

8.9 Gap 7 — No SHA256 Checksums on Release Assets

Severity: Medium

A search across all fifteen workflow files returns zero results for sha256sum being attached to a GitHub Release. publish_chocolatey.yml computes a per-file hash, but only for the Chocolatey registry — it never appears as a downloadable release asset.

Proposed Fix: A finalize job that runs after all platform build jobs complete, downloads all artifacts, generates SHA256SUMS.txt, and uploads it as a release asset.

8.10 Gap 8 — Docker Images Built But Not Distributed

Severity: Low–Medium

All three jobs in build_docker.yml use load: true, which loads into the local runner only and is discarded at job end. The push: true key is missing. The Dockerfile supports three production-grade build variants, each tested against --version. These are distribution artifacts that currently never reach any user.

Proposed Fix: Add release: types: [published] trigger, replace load: true with push: true, and add GHCR login. Requires adding packages: write permission — no additional repository secrets required.

8.11 Summary

Gap	Fix	Severity
Version triple desync	Pre-tag version sync PR + CI gate	Critical
Build provenance lost	CMake flag in CI build step	High
SHA256 self-check broken	Source fix — realpath()/GetModuleFileName()	Medium
No macOS release artifact	Trigger + upload step on build_shell job	High
Trigger mismatch	One-line change: created → published in release.yml	Medium
Unmaintained upload action	Migrate to softprops/action-gh-release@v2	Medium
No SHA256SUMS.txt	New finalize job	Medium
Docker not distributed	push: true + GHCR login + release trigger	Low–Medium

These changes preserve the existing modular workflow design while introducing two critical coordination mechanisms: (1) a mandatory pre-tag version synchronization step enforced by CI, and (2) a post-build aggregation step that ensures release completeness and integrity. Together, they eliminate version inconsistencies, restore platform parity, and make the CCExtractor 1.00 release pipeline deterministic, auditable, and production-ready.

---

9. Project Timeline and Deliverables

Week	Focus Area	Deliverables
Week 1–2	CI Reliability	Fix false positives, align status with UI
Week 3–4	Test Validation	Identify regressions vs improvements, update expected outputs
Week 5–6	Code Cleanup	Remove redundant C/Rust paths
Week 7–8	Test Coverage	Add missing coverage and required samples
Week 9–10	GUI Sync	Ensure Flutter GUI parity with CLI
Week 11–12	Release Pipeline	Improve automation and packaging
Final	Stabilization	All tests passing, documentation, final polish

---

Thank you for considering my proposal!

I would be excited to contribute to CCExtractor this summer and continue improving its reliability and release infrastructure.