Version: 1.0.0 **Target:** 0.119.0-Beta

Implementation Plan - Download All Reports Feature

This plan implements the backend logic and frontend interface to zip and download all generated reports for a specific session.

User Story

As a user, I want a single button on the dashboard to download all generated reports (charts, text, animations) as a ZIP archive, named with the year and contest, so I can easily archive the analysis.

Proposed Changes

1. `web_app/analyzer/views.py`

- **Import:** Add `shutil`, `FileResponse` from `django.http`, and `_sanitize_filename_part`.
- **New View:** Implement `download_all_reports(request, session_id)`.
  - **Context Loading:** Load `dashboard_context.json` to extract metadata safely without re-parsing files.
  - **Filename Construction:** Parse `report_url_path` from context to get `year` and `contest_name`.
  - **Archiving:** Use `shutil.make_archive` to zip the `reports/` directory located within the session storage.
  - **Streaming:** Return the zip file as a downloadable attachment.

2. `web_app/analyzer/urls.py`

- **Route:** Register `path('report/<str:session_id>/download_all/', views.download_all_reports, name='download_all_reports')`.

3. `web_app/analyzer/templates/analyzer/dashboard.html`

- **UI Update:** Locate the "Raw Data Access" card.
- **Button Activation:** Convert the disabled button into a live primary button pointing to the new URL.
- **Styling:** Ensure the button is "prominent" (Primary color) as requested.

Verification Plan

Automated Verification

- **Syntax Check:** The Builder will verify Python syntax.
- **Link Verification:** Verify the URL template tag generates a valid path.

Manual Verification Steps

1. Run a standard analysis (e.g., `test_code/Test_Logs/CQWW_CW`).
2. On the dashboard, scroll to the bottom.
3. Click "Download All Reports".
4. **Verify:** A file named like `2024_cq_ww_cw.zip` is downloaded.
5. **Verify:** Unzipping the file reveals the `reports/` directory structure containing `.png`, `.html`, and `.txt` files.

Surgical Changes

File: `web_app/analyzer/views.py`

Version: 0.117.1-Beta -> 0.119.0-Beta

___CODE_BLOCK___python

... existing imports ...

import shutil # Added from django.http import Http404, JsonResponse, FileResponse # Added FileResponse

... existing code ...

from contest_tools.reports._report_utils import _sanitize_filename_part

... existing code ...

def qso_dashboard(request, session_id): # ... existing implementation ... return render(request, 'analyzer/qso_dashboard.html', context)

def download_all_reports(request, session_id): """ Zips the entire 'reports' directory for the session and serves it as a download. Filename format: YYYY_CONTEST_NAME.zip """ session_path = os.path.join(settings.MEDIA_ROOT, 'sessions', session_id) context_path = os.path.join(session_path, 'dashboard_context.json')

```
if not os.path.exists(session_path) or not os.path.exists(context_path):
    raise Http404("Session or context not found")
```

```
# 1. Get Metadata for Filename
try:
    with open(context_path, 'r') as f:
        context = json.load(f)

    # report_url_path format: "YYYY/contest_name/event/calls"
    path_parts = context.get('report_url_path', '').split('/')
    if len(path_parts) >= 2:
        year = _sanitize_filename_part(path_parts[0])
        contest = _sanitize_filename_part(path_parts[1])
        zip_filename = f"{year}_{contest}.zip"
    else:
        zip_filename = "contest_reports.zip"
except Exception:
    zip_filename = "contest_reports.zip"

# 2. Create Zip Archive
# We zip 'reports' directory relative to session_path to avoid recursive loops
reports_root = os.path.join(session_path, 'reports')
if not os.path.exists(reports_root):
    raise Http404("No reports found to archive")

# Output zip path (stored in session root, not inside reports!)
zip_output_base = os.path.join(session_path, 'archive_temp')

try:
    # make_archive appends .zip automatically
    archive_path = shutil.make_archive(
        base_name=zip_output_base,
        format='zip',
        root_dir=session_path,
        base_dir='reports'
    )

    # 3. Serve File
    response = FileResponse(open(archive_path, 'rb'), as_attachment=True, filename=zip_filen
    return response
except Exception as e:
    logger.error(f"Failed to zip reports: {e}")
    raise Http404("Failed to generate archive")
```

CODE_BLOCK

File: `web_app/analyzer/urls.py`

Version: 0.114.0-Beta -> 0.119.0-Beta

___CODE_BLOCK___python

... existing imports ...

urlpatterns = [path('', views.home, name='home'), path('analyze/', views.analyze_logs, name='analyze'), path('analyze/progress/s t r : r e q u e s t _ id/', views.get_progress, name='get_progress'), path('report/s t r : s e s s i o n _ id/dashboard/', views.dashboard_view, name='dashboard_view'), path('report/s t r : s e s s i o n _ id/dashboard/qso/', views.qso_dashboard, name='qso_dashboard'), path('report/s t r : s e s s i o n _ id/download_all/', views.download_all_reports, name='download_all_reports'), # Added path('report/str:session_id/path:file_path', views.view_report, name='view_report'),] **CODE_BLOCK**

File: `web_app/analyzer/templates/analyzer/dashboard.html`

Version: N/A (Template)

___CODE_BLOCK___html