ChatGPT

Spectra App – Feature Review and Implementation Checklist

This document compares the original **spectra-app** repository with the current **spectra-app-beta** build. It outlines features present in the original application, highlights missing pieces in the beta build, and provides a checklist of tasks to bring the beta version to parity or beyond. The goal is to support multiple file types, remote data sources, robust data operations and a polished UI.

File Type Support

The original application uses dedicated ingestion modules to handle various spectral file formats:

Format	Key functions (original repo)	Required libraries	Beta status
ASCII/CSV/TXT	`parse_ascii_segments()` in `ingest_ascii.py` reads arbitrary delimited text, guesses wavelength & flux columns using header aliases, converts units, deduplicates wavelengths and builds downsample tiers [1]	`numpy`, `pandas`, `astropy.units`	Beta currently loads example CSVs but lacks automatic column detection, unit conversion and downsample tier generation.
FITS	`ingest_fits.py` reads 1-D or 3-D FITS files via `astropy.io.fits`, extracts wavelength and flux, handles event lists vs binned spectra, applies WCS if present, converts wavelength to nm and flux to canonical units, and builds downsample tiers [2] [3]	`astropy.io.fits`, `astropy.units`, `astropy.wcs`	Not implemented in beta.

Format	Key functions (original repo)	Required libraries	Beta status
JCAMP-DX (IR)	`ingest_jcamp.py` (not fully viewed) parses JCAMP-DX infrared files, extracts wavenumber spectra, converts to wavelength and transforms transmittance/absorbance using helper functions in `ir_units.py` [4].	`numpy`, `bs4` (for HTML), custom IR unit logic	Not implemented in beta.
Remote data from archives	Fetchers under `app/server/fetchers` connect to external services:		
• **NIST ASD** – `fetchers/nist.py` uses `astroquery.nist` to obtain spectral line lists for a selected element or linename; converts wavelengths to nm and normalizes intensities [5].			
• **ESO X-Shooter** – `fetchers/eso.py` downloads curated FITS spectra from Zenodo, parses them with `astropy.io.fits` and returns wavelength and flux arrays with metadata [6].			
• **SDSS DR17** – `fetchers/sdss.py` retrieves Sloan spectra via HTTP, reads FITS files and converts to canonical units [7].			

Format	Key functions (original repo)	Required libraries	Beta status
• **NIST Quant IR** – `fetchers/nist_quant_ir.py` scrapes the NIST Quantitative IR webbook, fetches JCAMP files, parses them and converts to nm [8].	`astroquery`, `requests`, `astropy`, `beautifulsoup4`	None of these remote fetchers are exposed in the beta UI.	

Action items:

- [ ] Create ingestion layer in beta that detects file extension and calls the appropriate parser:
- `.csv` / `.txt` / `.tsv` → ASCII parser.
- `.fits` / `.fit` → FITS parser.
- `.jdx` / `.dx` → JCAMP parser.
- Unknown → prompt user or attempt simple CSV parse.
- [ ] Add optional dependency guards: if `astropy` or `pandas` is missing, show an informative error.
- [ ] Integrate remote fetchers into a “Data → Download” menu. Provide search fields for element/linename (NIST), target names (ESO/SDSS), etc. Display metadata and allow users to import the fetched spectra.

Data Operations

Operation	Description (original repo)	Beta status
Resample to common grid	`resample_to_common_grid()` in `differential.py` interpolates two spectra onto a common wavelength grid before computing arithmetic [9].	Not yet implemented in beta; subtract/ratio operations would currently mismatch wavelength grids.
Difference and ratio	Functions `subtract()` and `ratio()` compute point-wise A–B and A/B operations with epsilon guarding [10].	Not exposed in beta UI.
Downsample tiers	`utils/downsample.py` provides LTTB and min-max envelope algorithms to generate multiple downsample tiers for efficient zooming and panning [11].	Beta uses a simple peak downsampler but does not build multiple tiers nor refresh LOD on zoom.

Operation	Description (original repo)	Beta status
Unit conversion	In ingestion modules, wavelength units (nm, Å, µm, cm^{-1}) and flux units are parsed with `astropy.units` and converted to canonical units. IR units are transformed using `ir_units.py` [4].	Beta can toggle display units but always stores x values in nm; does not yet convert on import, nor does it track original units in metadata.
Metadata & provenance	Ingest functions assemble a rich `metadata` dictionary with instrument, telescope, date, etc., using header alias maps and a `provenance` log describing unit conversions and other transformations [12].	Beta displays a few fields (name, source, x/y ranges) but does not show original metadata or a provenance log.
Normalization & smoothing	Original UI (not reviewed) allowed normalizing spectra (max/area) and applying Savitzky–Golay smoothing.	Toolbar in beta has “Normalize” and “Smoothing” comboboxes but they are not yet functional.

Action items:

- [ ] Implement resampling to common grid before math operations. Use `numpy.interp()` as in the original `resample_to_common_grid` to ensure overlapping wavelength ranges.
- [ ] Expose subtraction and division in the Math tab. Allow the user to choose which two traces are A and B, select operation, and add result as a new derived trace.
- [ ] Replace the simple downsampler with tiered downsampling (`build_downsample_tiers`) and refresh curves on zoom. Use the min-max envelope for coarse tiers (≤2000 points) and LTTB for finer tiers [13].
- [ ] On ingestion, parse the original file’s wavelength and flux units with `astropy.units` and convert to nm and canonical flux units. Preserve original unit labels in `metadata`.
- [ ] Record provenance: store a list of transformation steps (unit conversion, deduplication, normalization, math operation) and display this in the Provenance tab.
- [ ] Activate normalization and smoothing controls in the toolbar. Use `numpy`/ `scipy.signal.savgol_filter` for smoothing.

User Interface & Workflow

Feature	Expectation from original app	Beta status	Notes
Dataset Browser	Tree groups for “Originals” and “Derived” with color swatches, checkboxes for visibility and context menu for remove/rename.	Present in beta (left dock).	Good foundation; need context menu actions and support for grouping derived traces.

Feature	Expectation from original app	Beta status	Notes
Plotting	Use pyqtgraph with interactive pan/zoom, crosshair, legend, unit-toggle; handle millions of points via downsample tiers; overlay markers for spectral lines.	Beta shows multi-trace plot with crosshair and status readout; uses peak downsample and no overlay support.	Need dynamic LOD, overlay of line lists, normalization and smoothing.
Inspector tabs	Tabs for Info (metadata), Math (operations), Style (color & smoothing), Provenance (JSON diff).	Present in beta but mostly placeholders.	Populate tabs with metadata from ingestion, math controls, style editors and provenance logs.
Data table	Show raw x/y values for selected trace(s).	Beta displays table but always shows first loaded sample.	Should update to reflect the currently selected dataset.
Remote fetchers UI	Provide UI to search and fetch spectra from NIST, ESO, SDSS, MAST, etc.	Not available.	Add "Import → Remote Data" dialog with search fields and result list.
File import	Drag-and-drop or file picker with multiple file type support.	Beta can open CSV examples but file types are limited.	Extend to accept fits/jcamp and remote fetcher results; show progress while parsing large files.
Export/Save	Export plot as image and selected datasets as CSV + manifest (metadata/provenance).	Beta has an Export button but no implementation.	Use pyqtgraph's `ImageExporter` and write out CSV plus JSON for metadata.
Keyboard shortcuts & gestures	Ctrl+O to open, numeric keys to change render mode, toggles for crosshair/legend.	Partially present in beta.	Expand to include shortcuts for normalization (`N`), unit cycling (`U`), math operations, etc.

Libraries & Dependencies

The original repository relies on several scientific libraries:

- **Astropy** – FITS I/O, unit handling, WCS. Used for FITS ingestion and unit conversions [2] [3].
- **Pandas** – Parsing ASCII/CSV files with flexible delimiter and numeric coercion in `ingest_ascii.py` [1].

- **Astroquery** – Accessing NIST line lists (`astroquery.nist`), MAST, etc. Needed for remote fetchers [14].
- **Requests** – Downloading remote FITS/JCAMP files (ESO, SDSS, NIST Quant IR) [6] [15].
- **BeautifulSoup** – Parsing HTML catalogs for NIST Quant IR [8].
- **NumPy/SciPy** – Arrays, interpolation, Savitzky–Golay smoothing, downsample algorithms.
- **PyQtGraph** – High-performance plotting. Beta already uses this.

Ensure these dependencies are added to `requirements.txt` and handle optional imports gracefully (e.g., remote fetchers should fail gracefully if `astroquery` is missing).

Implementation Checklist

To upgrade **spectra-app-beta**, follow this checklist:

1. **Ingestion layer**
2. [ ] Create `ingest.py` in beta that routes based on file extension and calls CSV, FITS or JCAMP parser.
3. [ ] Port or reimplement `parse_ascii_segments()` to support header aliases, unit detection and downsample tiers.
4. [ ] Port or reimplement FITS ingestion using `astropy.io.fits` and unit conversions to nm.
5. [ ] Port JCAMP parser and IR unit conversion; use `ir_units.py` to convert transmittance to absorbance.
6. [ ] Populate `metadata` and `provenance` objects during ingestion.
7. **Remote fetchers**
8. [ ] Add UI dialog to search and fetch data from NIST, ESO, SDSS, MAST and NIST Quant IR. Use `astroquery` or `requests` as appropriate.
9. [ ] Cache downloaded data in a local directory and record SHA-256 checksums and fetch timestamps.
10. **Math & resampling operations**
11. [ ] Implement resampling to common grid before subtraction or ratio. Use `numpy.interp()` as in original `differential.py` [9].
12. [ ] Add Math tab controls to select traces A and B, choose operation (A–B or A/B) and create derived trace.
13. [ ] Provide epsilon handling for division and update metadata/provenance accordingly.
14. **Downsampling & performance**
15. [ ] Replace custom peak downsampler with `build_downsample_tiers()` from `downsample.py` [13].
16. [ ] Regenerate appropriate tier when the user zooms; default tiers could be 500/2000/8000 points.
17. [ ] Keep original high-resolution data for exports and math operations.

18. **UI improvements**

19. [ ] Populate Info tab with metadata fields (instrument, telescope, exposure time, etc.).
20. [ ] Fill Provenance tab with a tree of transformations (unit conversion, resampling, normalization, math operations). Provide JSON view.
21. [ ] Implement Style tab controls for color, line width, smoothing window and normalization. Apply changes immediately.
22. [ ] Update Data table to reflect currently selected trace; allow export of raw values.
23. [ ] Add context menu to dataset tree for rename/duplicate/delete/hide.

24. [ ] Add overlay capability: fetch spectral lines (e.g., NIST) and show as vertical markers with labels.

25. **Export & persistence**

26. [ ] Implement exporting of plots to PNG and datasets to CSV/JSON with metadata and provenance.

27. [ ] Save and restore user sessions (open files, derived traces) using a project file format (optional).

28. **Error handling & messaging**

29. [ ] Catch exceptions during ingestion or fetch; show user-friendly messages.
30. [ ] Warn when units are ambiguous or unsupported; allow user to override.

31. [ ] Handle missing dependencies gracefully; e.g., disable NIST fetcher if `astroquery` is unavailable.

32. **Testing**

33. [ ] Write unit tests for ingestion functions using sample data.
34. [ ] Test remote fetchers against known IDs.
35. [ ] Ensure math operations handle non-overlapping wavelength ranges correctly.
36. [ ] Verify UI remains responsive when loading million-point spectra.

By following this checklist, the **spectra-app-beta** will gain full parity with the original application's ingestion capabilities, robust mathematical operations, remote data integration and a rich user interface.

---

1 12 spectra-app/app/server/ingest_ascii.py at main · brettadin/spectra-app · GitHub
https://github.com/brettadin/spectra-app/blob/main/app/server/ingest_ascii.py

2 spectra-app/app/server/ingest_fits.py at main · brettadin/spectra-app · GitHub
https://github.com/brettadin/spectra-app/blob/main/app/server/ingest_fits.py

3 spectra-app/app/server/fetchers/nist.py at main · brettadin/spectra-app · GitHub
https://github.com/brettadin/spectra-app/blob/main/app/server/fetchers/nist.py

4 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/ir_units.py

5 14 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/fetchers/nist.py

6 15 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/fetchers/eso.py

7 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/fetchers/sdss.py

8 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/fetchers/nist_quant_ir.py

9 10 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/server/differential.py

11 13 raw.githubusercontent.com
https://raw.githubusercontent.com/brettadin/spectra-app/main/app/utils/downsample.py