# Comprehensive Review of Spectra App (brettadin/spectrasuite)

## Application Overview

Spectra App is a research‑grade **Streamlit** application that allows astronomers to upload spectra, compare them against archival references and export reproducible bundles. According to the README, the app implements a canonical wavelength baseline, robust ASCII/FITS ingestion, a trace manager that surfaces axis classifications and detection methods, and an export pipeline that bundles manifests and trace CSVs[[1]](https://github.com/brettadin/spectrasuite/blob/main/README.md#L1-L11). The project is organised into several directories:

* **app/ui** – Streamlit UI tabs (Overlay, Differential, Star Hub, Docs). The UI contract notes that the sidebar contains controls for examples, display mode, units, duplicate scope and line overlays[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12). The overlay tab provides a file uploader, trace manager with axis family captions and transform provenance, Plotly chart, and provenance caption[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12). The Star Hub tab currently contains a SIMBAD resolver placeholder[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12).
* **server/ingest** – ingestion functions. ASCII ingestion normalises column names, detects wavelength/flux columns, falls back to unit hints or numeric heuristics and records provenance[[3]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L3-L38). FITS ingestion extracts metadata, normalises to vacuum nanometres and logs provenance.
* **server/overlays** – line overlay utilities (e.g., Fe I catalogue). The overlay UI allows users to select a species and apply velocity shifts; it scales lines and converts units via server/math/transforms[[4]](https://github.com/brettadin/spectrasuite/blob/main/app/ui/overlay.py#L161-L180).
* **atlas/ and brains/** – internal documentation and run journals. For example, v1.0.0j introduced axis‑summary captions that display the axis family, unit, detection method and headerless status under each trace[[5]](https://github.com/brettadin/spectrasuite/blob/main/brains/v1.0.0j__assistant__axis_provenance_ui.md#L9-L22). The architecture notes summarise that the front‑end is a Streamlit app with modular tab renderers, while the server layer handles ingestion, maths and export[[6]](https://github.com/brettadin/spectrasuite/blob/main/atlas/architecture.md#L3-L10).
* **PATCH\_NOTES** – user‑facing change logs. Version 1.0.0(k) added transform provenance notes (unit conversions, air–vacuum corrections, differential operations) to the overlay trace manager[[7]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L2-L15) and identified stubbed archive fetchers and replay tooling as known issues[[8]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L17-L21).

## File Upload and Data Display

### Upload Support

* **ASCII ingestion** uses load\_ascii\_spectrum (pandas) to autodetect delimiters, re‑imports without headers when the first row is numeric, canonicalises column names, and detects wavelength/flux columns through aliases, unit hints or numeric heuristics[[9]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L3-L36). It handles frequency/energy units by converting them to wavelength and logs whether detection relied on aliases, unit hints or heuristics[[10]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L16-L37). Rows without finite wavelength/flux values are filtered, and a SHA‑256 hash is stored for deduplication[[11]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L20-L34).
* **FITS ingestion** (v0.1.0 b) added header/WCS parsing, uncertainty support, and unit normalisation. A SHA‑256 hash prevents duplicate ingest and the overlay uploader accepts FITS files alongside ASCII uploads[[12]](https://github.com/brettadin/spectrasuite/blob/main/ai.patch#L130-L145). Wavelength units are normalised to the vacuum‑nm baseline for consistent comparisons.
* **Provenance logging** records ingestion parameters (e.g., axis family, detection method, headerless flag) in ProvenanceEvent. The overlay trace manager displays this information with axis‑family captions and transform notes[[5]](https://github.com/brettadin/spectrasuite/blob/main/brains/v1.0.0j__assistant__axis_provenance_ui.md#L9-L22). Version 1.0.0 k extended this to summarise downstream transforms such as unit conversions, air‑to‑vacuum corrections and differential operations[[7]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L2-L15).

The ingestion pipelines should handle typical ASCII and FITS files correctly. However, large datasets could strain Streamlit’s single‑threaded execution and the client‑side memory used by Plotly. The code uses pandas and numpy arrays; for very large spectra (>100k points or dozens of traces) this could lead to sluggish plotting and high memory usage. To improve robustness:

* **Progress indicators:** Wrap ingestion in st.spinner and display progress bars when reading large files, to give users feedback during heavy operations.
* **Downsampling/decimation:** Offer an optional downsampling step (e.g., using scipy.signal.decimate) to reduce the number of points plotted without losing overall shape. Users could choose between full resolution and decimated views.
* **Asynchronous loading:** Streamlit recently added @st.experimental\_async—ingestion and heavy transforms can be executed asynchronously to keep the UI responsive.

### Data Display and Accuracy

The overlay tab uses Plotly to render traces and allows toggling visibility, axis units and line overlays. The axis‑summary captions introduced in v1.0.0j show the axis family (wavelength, wavenumber, frequency or energy), detection method and unit[[5]](https://github.com/brettadin/spectrasuite/blob/main/brains/v1.0.0j__assistant__axis_provenance_ui.md#L9-L22). Transform provenance notes summarise unit conversions, air–vacuum corrections and differential steps[[7]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L2-L15). These features help verify that data is displayed correctly and that conversions are transparent.

Potential issues to watch for:

* **Axis misclassification:** Although alias matching and unit hints cover many cases[[13]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L9-L18), unusual column names or ambiguous units could still misclassify the axis. Encourage users to review the axis caption; if misclassified, provide a manual override (currently absent) to specify the correct axis family.
* **Headerless detection:** The ingestion pipeline re‑imports headerless files but may still mis‑detect if both columns are numeric. The provenance caption now shows when numeric heuristics fired[[14]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L9-L21); consider surfacing a warning and allowing the user to manually choose wavelength/flux columns.
* **Uncertainty/error bars:** Spectra often include uncertainty arrays. FITS ingestion captures these but the UI currently displays only flux; adding an option to display shaded error regions would enhance analysis.

### Fetching External Data

The Star Hub tab integrates a SIMBAD resolver but external archive fetchers (MAST/SDSS) are stubbed[[8]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L17-L21). To fetch archival spectra, implement adapters using astroquery to query MAST or SDSS and ingest returned FITS files through the existing pipeline. Provide caching and asynchronous loading to avoid blocking the UI.

## Site Performance and Scalability

Streamlit’s architecture is inherently single‑threaded and can become sluggish with heavy computations or many simultaneous users. Plotly traces with tens of thousands of points can also slow down browser rendering. Recommendations:

* **Limit number of concurrent traces:** Provide a warning or disable new uploads after a configurable threshold (e.g., 5–10 traces). Alternatively, collapse older traces in the UI and render only a summary preview until expanded.
* **Use session state wisely:** The AppSessionState stores traces and dedup ledger[[6]](https://github.com/brettadin/spectrasuite/blob/main/atlas/architecture.md#L3-L10). Remove unused traces from session state when the user unchecks them or exports, to free memory.
* **Chunked streaming:** For extremely long spectra, send data in chunks to Plotly via streamlit.delta\_generator.add\_rows, though this is more suited to time‑series streams than static spectra.
* **Server deployment:** When deploying, use Streamlit’s --server.maxUploadSize and --server.maxMessageSize settings to control memory usage. For multi‑user scenarios, run behind a reverse proxy with multiple worker processes (e.g., using gunicorn via streamlit-on-fastapi), since standard Streamlit cannot scale horizontally.

## UI and UX Improvements

The UI largely follows the contract defined in atlas/ui\_contract.md[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12), but several enhancements could improve usability:

1. **Clutter reduction:** The sidebar currently lists example data, display mode, units, duplicate scope and line overlay controls[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12). Group related controls into collapsible sections or accordions (e.g., “Examples & Duplicates,” “Display Settings,” “Line Overlays”). Collapsible panels reduce visual clutter while preserving functionality.
2. **Duplicate toggle:** The session ledger stores a dedup signature but the “Allow duplicates” flag in the sidebar is not yet implemented[[15]](https://github.com/brettadin/spectrasuite/blob/main/atlas/data_model.md#L8-L11). Expose this flag to users with a clear explanation (e.g., “Allow re‑uploading identical files”) and implement the logic to override deduplication when enabled.
3. **Manual axis override:** Add drop‑down selectors in the overlay tab that let users override the automatically detected axis family and units. This helps when the heuristics misclassify unusual data.
4. **Error‑bar display:** Provide an option to show uncertainty arrays as shaded regions or error bars. This could be toggled in the trace manager.
5. **Theme consistency:** Use consistent colour schemes and fonts. Streamlit 1.30+ allows theme settings in .streamlit/config.toml. Choose a dark or light theme that matches the domain (astronomy) and apply to Plotly charts for better contrast.
6. **Star Hub expansion:** Replace the placeholder with actual integrations to archives (MAST, SDSS) and include quick‑look plots for fetched spectra. Provide search history and caching to improve repeat queries.
7. **Docs tab enrichment:** The docs tab renders markdown from docs/static (currently a placeholder). Populate it with tutorials (e.g., uploading examples, interpreting axis provenance), explanation of differential analyses, and FAQs. Link to atlas/brains/patch notes for deeper reading.

## Codebase Clean‑Up and Redundancies

The repository is relatively well organised. Running ruff, black and mypy is part of CI. However, a few areas could be streamlined:

* **Unused allow duplicates flag:** As noted in the data model, the duplicate‑scope flag is “unused yet”[[15]](https://github.com/brettadin/spectrasuite/blob/main/atlas/data_model.md#L8-L11). Remove it from the UI until implemented or finish implementing the logic.
* **Stubbed archive adapters:** The ingestion adapters for MAST and SDSS are stubbed and appear in patch notes as known issues[[8]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L17-L21). Remove unused placeholder code or fill in proper implementations.
* **Large ai.patch file:** The repository contains an ai.patch diff containing the entire project history[[16]](https://github.com/brettadin/spectrasuite/blob/main/ai.patch#L1-L94). This patch file is likely a leftover from initial scaffolding and should be excluded from the package distribution. Move it to documentation or remove it to reduce repository size.
* **Simplify tests and verifiers:** The tools/verifiers scripts enforce updates to atlas/brains/patch notes/handoff and the UI contract. While useful for continuous integration, they may be overkill for local development. Document their usage and provide a way to bypass them in quick development iterations.

## Documentation Updates

The documentation is spread across several directories: atlas, brains, PATCH\_NOTES and handoffs. To make it more approachable:

* **Consolidate docs:** Provide a high‑level index in docs/static that links to atlas (architecture and data model notes), patch notes (user‑facing change log), brains (development journal) and handoff documents. Users should know where to find design details versus change logs.
* **Version badges:** Display the current app version (e.g., 1.0.0k) in the header, linking to the corresponding patch notes[[17]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L5-L7). This aids discoverability of new features and known issues.
* **Tutorials and FAQs:** Add step‑by‑step tutorials for uploading ASCII and FITS files, performing differential analyses and exporting results. Include troubleshooting tips for common ingestion errors (e.g., mislabelled columns) and explanation of axis provenance semantics.
* **API reference:** Document the Python functions (load\_ascii\_spectrum, canonicalize\_ascii, load\_fits\_spectrum, etc.), expected inputs/outputs, and examples. This is valuable for developers who wish to integrate the ingestion pipeline outside the Streamlit UI.

## Conclusion

The Spectra App is a promising spectral analysis tool with a well‑documented ingestion pipeline and transparent provenance. The documentation (atlas, brains and patch notes) shows a commitment to reproducibility and auditability. To improve the app further, focus on enhancing the UI (reducing clutter, adding manual overrides and error‑bar display), implementing asynchronous loading and downsampling for performance, fleshing out the Star Hub and docs content, and cleaning up unused flags and placeholder files. Updating the documentation to reflect these changes and consolidating information will help users and developers alike.

[[1]](https://github.com/brettadin/spectrasuite/blob/main/README.md#L1-L11) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/README.md>

[[2]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L3-L12) [[17]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md#L5-L7) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/atlas/ui_contract.md>

[[3]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L3-L38) [[9]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L3-L36) [[10]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L16-L37) [[11]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L20-L34) [[13]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L9-L18) [[14]](https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md#L9-L21) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/atlas/ingest_ascii.md>

[[4]](https://github.com/brettadin/spectrasuite/blob/main/app/ui/overlay.py#L161-L180) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/app/ui/overlay.py>

[[5]](https://github.com/brettadin/spectrasuite/blob/main/brains/v1.0.0j__assistant__axis_provenance_ui.md#L9-L22) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/brains/v1.0.0j__assistant__axis_provenance_ui.md>

[[6]](https://github.com/brettadin/spectrasuite/blob/main/atlas/architecture.md#L3-L10) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/atlas/architecture.md>

[[7]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L2-L15) [[8]](https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md#L17-L21) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/PATCH_NOTES/PATCH_NOTES_v1.0.0(k).md>

[[12]](https://github.com/brettadin/spectrasuite/blob/main/ai.patch#L130-L145) [[16]](https://github.com/brettadin/spectrasuite/blob/main/ai.patch#L1-L94) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/ai.patch>

[[15]](https://github.com/brettadin/spectrasuite/blob/main/atlas/data_model.md#L8-L11) GitHub

<https://github.com/brettadin/spectrasuite/blob/main/atlas/data_model.md>