Project Report: Athena Internal Learning Dashboard

Candidate: Farhan Abid Ahmed

Date: 26th June, 2025

Repository Link: FarhanDipto/Athena-Case-Study

1. Project Overview

This project delivers a comprehensive internal dashboard for Athena, a student-centered learning platform. Built using Retool, the dashboard streamlines the process of collecting, reviewing, and analyzing student-submitted concept explanations, which are a core part of Athena's Feynman Technique-based methodology.

The final application consists of three primary views, organized within a clean, tabbed interface:

- **Student Submission Panel:** A user-friendly form for students to submit their explanations, including optional voice note uploads.
- **Admin Review Dashboard:** A powerful, interactive table for staff to manage, filter, rate, and tag all submissions. It includes features for in-line editing and automatic flagging of duplicate topic submissions by a student.
- **Analytics Dashboard:** A visual summary of platform engagement, featuring charts for the most submitted topics, top student contributors, and a live count of submissions pending review.

2. Implementation Decisions & Technology Choices

Several key decisions were made to ensure rapid development, maintainability, and a robust user experience within the case study's constraints.

- **Frontend/Tooling (Retool):** Retool was chosen as specified, leveraging its powerful UI components (Table, Form, Chart, Containers) and event-driven architecture to build a feature-rich interactive application in a short timeframe.

- **Backend (Google Sheets):** For the purpose of this case study, Google Sheets was selected as the backend data store.
    - **Reasoning:** Its seamless, native integration with Retool allows for rapid prototyping and deployment without database setup overhead. It provides a lightweight, persistent, and easily auditable data source perfect for this project's scope.

- **Data Structure:** A structured schema was designed in Google Sheets, including a unique submissionId for each entry. This was critical for enabling the "Update a spreadsheet" functionality, as it provides a reliable key to filter and update specific rows.
- **Application Architecture:**
    - A form is place for students to submit their text queries and voice notes to the admins, who then can see the textual queries and voice notes (the implementation is there, but not available now for cloud constraint), and decide their review status, flag them with review tags, and rate them between 5 stars.
    - **Data Flow:** A primary query (getSubmissions) fetches all data. A secondary "Query JSON with SQL" query (transformSubmissions) is then used to apply filters non-destructively, ensuring the UI is fast and responsive.

3. Challenges Faced & Solutions

During development, several technical challenges were encountered and successfully resolved.

- **Challenge 1: In-line Table Edits Wiping Unchanged Data.**
    - **Problem:** When saving an edit to a single cell in the admin table (e.g., changing a rating), the updateSubmission query was overwriting other editable columns in the same row with blank values.
    - **Solution:** The update query's logic was made more robust. Instead of just sending the value from the changesetArray, a ternary operator was implemented: {{ table.changesetArray[0].fieldName !== undefined ? table.changesetArray[0].fieldName : table.selectedRow.fieldName }}. This ensures that the query sends the newly edited value for the changed field, while sending the existing selectedRow data for all other fields, preventing data loss.
- **Challenge 2: Form Submission Adding Blank Rows.**
    - **Problem:** The student submission form was successfully appending a new row to Google Sheets, but the row contained no data.
    - **Solution:** This was a scope and naming issue. By inspecting the form's state (form.data object), the exact component names (e.g., textInput1, emailInput1) were identified. The addSubmission query was updated to reference these precise keys, ensuring a perfect mapping between the form fields and the database columns.
- **Challenge 3: Implementing Duplicate Submission Flagging.**
    - **Problem:** Reliably flagging a submission as a duplicate required checking against the entire dataset, not just the filtered view.
    - **Solution:** A "Custom Column" was added to the admin table. Its value is calculated using a JavaScript snippet that filters the master getSubmissions.data

array for entries where both the studentEmail and topic match the currentRow. If the resulting array's length is greater than one, it flags the row as a duplicate.

- **Challenge 4: Legacy Function's Unavailability Due To Retool's Update.**
    - **Problem:** Many basic functions that were there before, has been changed/moved/removed completely from the app. The biggest issue of this problem was, "Key-value pair" functionality was changed to something else completely. As a result, admin changed/edits were not being saved properly.
    - **Solution:** Custom JS scripting was used in such cases to assign correct resource values to correct keys. The keys then acted as APIs that updated the correctly fetched changes to the specified rows in the designated columns.

4. Potential Future Improvements

If this project were to move into a full production environment, I would recommend the following enhancements:

- **Migrate to a Production Database:** Transition the backend from Google Sheets to a scalable SQL database like PostgreSQL. This would offer improved performance, data integrity, and scalability.
- **Robust User Authentication:** Implement a proper login system (e.g., via Retool's native auth or an external provider like Auth0) to securely distinguish between student and admin roles, showing the appropriate views automatically.
- **Direct File Uploads:** Integrate the File Picker (for voice notes) component with a cloud storage solution like Amazon S3. This would allow for the direct upload of voice notes, storing the secure file URL in the database instead of just the filename.
- **Advanced Analytics:** Introduce date-range filters on the Analytics Dashboard to allow staff to view engagement trends over specific periods.