Daily Technical Report (11-02-2026)

📌 Executive Summary

Today's development focused on three major areas:

1. **Refining Program Selection Logic**: Ensuring robust API parameter handling for the external search service.
2. **Implementing "Multiple University Applications" System**: A comprehensive full-stack feature allowing students to apply to multiple programs, with tracking for Agents and Admins.
3. **UI/UX Enhancements & Optimization**: Polishing the student dashboard, implementing modals, and cleaning up backend scripts.

---

1. Feature: Refined Program Selection Flow

Objective: Optimize the API calls for program fetching to support optional specialization filters.

- **Frontend (`ProgramSelectionFlow.jsx`)**: Updated logic to treat "Specialization" as an optional parameter. The "Proceed" button is now enabled even if no specialization is selected.
- **Backend Support**: Verified `fetchPrograms` service to correctly construct URL parameters, omitting `null` values to ensure accurate search results when filters are partial.

2. Feature: Multiple University Applications System (Full Stack)

Objective: Enable agents to manage multiple applications per student and provide visibility into these applications across the platform.

Backend Implementation:

- **Controller Logic (`applicationController.js`)**:
  - **Refined Filtering**: Removed restrictive `$match` stages in `getPendingStudents` to allow existing applicants to appear in search results for re-application.

    - **Duplicate Prevention**: Implemented a guard clause `Application.findOne({ studentId, programId })` to prevent double-applying to the exact same program.
- **Data Aggregation (`studentController.js`)**:
    - Updated `getAllStudents` with an aggregation pipeline to calculate and return `applicationCount` for each student.
    - Enhanced `getStudentById` to populate the full list of applications, enabling detailed profile views.

Frontend Implementation:

- **Student List (`StudentList.jsx`)**:
    - Added a dynamic **"Applied"** column with color-coded badges (Green for "Applied", Gray for "No").
    - Displayed total application counts per student.
- **Student Details (`StudentDetails.jsx`)**:
    - Added a new **"Applied Programs"** section to the student profile.
    - Integrated `ProgramDetailsModal` for viewing application specifics without navigating away.
- **Modal Component (`ProgramDetailsModal.jsx`)**:
    - Created a reusable, animated modal using `framer-motion` to display comprehensive program details (University, Country, Tuition Fee, Intake).

3. Feature: Student "My Applications" Dashboard

Objective: Provide students with a dedicated, polished view of their own applications.

- **New Page (`StudentApplications.jsx`)**: Created a dedicated route `/my-applications`.
- **UI Enhancements**:
    - **Grid Layout**: Implemented a responsive **2-column grid (`md:grid-cols-2`)** for application cards.
    - **Navigation**: Added a functional **Back Button** (ArrowLeft) for easy dashboard return.
    - **Stats**: Added a dynamic **Total Count** badge in the header.
    - **Styling**: Applied consistent card styling and status badges (`getStatusColor`) to match the main dashboard.
- **Integration**:
    - Updated `Sidebar.jsx` to link "Applied Colleges" to this new page.
    - Updated `StudentDashboard.jsx` to map navigation correctly.

4. Codebase Maintenance & Cleanup

Objective: ensure project hygiene by removing obsolete files.

- **Script Cleanup**:
  - Deleted `backend/src/scripts/reset_student_password.js` (one-off utility).
  - Refactored `backend/src/scripts/seed.js` : Removed hardcoded dummy data (Universities, Courses) to prevent conflicts, keeping only Super Admin initialization.
- **Analysis**: Identified unused documentation (`.md` reports) and frontend assets (`react.svg` , `App.css`) for upcoming deletion.

Status: All features tested and verified. UI enhancements for the Student Application page have been approved.