

Full Stack Project: Online Student Feedback & Resolution System

Subject Code: 23CSP-339 Student: Ronak Jain (23BCS10225)

Module 3: Student Dashboard & Feedback Tracking

1. Objective

With the "CREATE" functionality implemented in Module 2, the objective of Module 3 was to build the corresponding "READ" functionality for students. The goal was to provide a secure, personal dashboard where a logged-in student can view a complete history of their own submissions, track the status of each feedback item (e.g., "open", "in_progress", "resolved"), and read any replies posted by the administration. This module is crucial for providing transparency and "closing the loop" for the student.

2. Technologies Used

- **Backend:** Spring Boot, Spring Security, Spring Data MongoDB
- **Frontend:** React (using `useEffect` and `useState` for data fetching), Axios, React Router, React Components
- **Database:** MongoDB

3. Backend Implementation (Spring Boot)

The backend was enhanced to securely provide *only* the logged-in student's data.

3.1. Repository (`repository` package)

- **`FeedbackRepository.java`:** This interface was already prepared in Module 2 with the required custom method:
 - `List<Feedback> findByStudentId(String studentId)`: Spring Data MongoDB automatically implements this method, generating a query to find all documents in the `feedback` collection that match the provided `studentId`.

3.2. Service Layer (`service` package)

- **`FeedbackService.java`:** A new method was added to act as the business logic layer for this feature:

- `getFeedbackByStudentId(String studentId)`: This method simply calls `feedbackRepository.findById(studentId)`, acting as a clean pass-through from the controller to the repository.

3.3. API Endpoint (**controller** package)

- **FeedbackController.java**: This controller was updated to add a new, secure endpoint for students:
 - `GET /api/feedback/my-feedback`: This endpoint is protected by Spring Security, requiring an authenticated user.
 - It uses the injected `Authentication` object to get the logged-in user's email.
 - It queries the `UserRepository` to find the user's unique MongoDB `_id` (the `studentId`).
 - This `studentId` is then passed to the `FeedbackService` to fetch the feedback list.
 - This design is inherently secure, as it is impossible for one student to request another student's data; the ID is always derived from their own authentication token.

4. Frontend Implementation (React)

The frontend's `StudentDashboard` was upgraded from a simple placeholder to a dynamic, data-driven page.

4.1. Service Layer (**services** package)

- **FeedbackService.js**: This service was updated with a new function:
 - `getMyFeedback()`: This function calls the new `GET /api/feedback/my-feedback` endpoint. It uses the `authHeader()` helper function (from Module 2) to attach the student's JWT, proving their identity to the backend.

4.2. UI Components (**components** package)

- **FeedbackList.js (New Component)**: A new component was created to be responsible *only* for rendering the list of feedback.
 - It receives the `feedbackItems` as a prop from its parent (`StudentDashboard`).
 - It maps over this array and renders each feedback item as a "card."
 - Each card displays the key information: `status` (with a color-coded tag), `category`, `rating` (as stars), the original `content`, and the submission `createdAt` date.

- **(Update from Module 5):** This component was later enhanced to also render the `item.thread` array, displaying any admin replies in a "Conversation Thread" section within the card.
- **FeedbackList.css (New File):** A dedicated CSS file was created to style the feedback cards, status tags, and the (later added) conversation thread, ensuring a clean and readable layout.
- **StudentDashboard.js (Updated):** This component was significantly rebuilt to manage the data-fetching lifecycle.
 - It now uses `useState` to hold the `feedbackList`, `loading`, and `error` states.
 - A `useEffect` hook is used to call `FeedbackService.getMyFeedback()` as soon as the component loads.
 - While fetching, it displays a "Loading feedback..." message. If an error occurs, it displays the error.
 - Once the data is successfully fetched, it is sorted by date and passed as a prop to the `FeedbackList` component for rendering.
 - A dedicated CSS file, `StudentDashboard.css`, was also added to give the dashboard a professional header and layout, replacing all previous inline styles.