Full Stack Project: Online Student Feedback & Resolution System

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

Module 2: Student Feedback Submission

1. Objective

Following the establishment of a secure authentication system in Module 1, the objective of Module 2 was to build the core "CREATE" functionality of the application. This module empowers authenticated students to submit structured, detailed feedback through a secure, user-friendly form. The system was designed to capture rich data (text, ratings, categories) and to respect the student's choice to submit feedback anonymously.

2. Technologies Used

- **Backend:** Spring Boot, Spring Data MongoDB, Spring Security
- **Frontend:** React (using `useState` for form management), Axios, React Router
- **Database:** MongoDB

3. Backend Implementation (Spring Boot)

The backend was expanded to securely receive, validate, and store feedback submissions.

3.1. Data Models (`model` package)

The non-relational nature of MongoDB was leveraged to create a highly efficient, self-contained data model that avoids complex database joins.

- **`Comment.java` & `ResolutionLog.java`:** Created as simple POJOs (Plain Old Java Objects) using Lombok (`@Data`). These classes are *not* separate documents; they are designed to be *embedded* within the main `Feedback` document, perfectly aligning with the NoSQL strategy.
- **`Feedback.java`:** This is the primary data model for this module, annotated with `@Document(collection = "feedback")`. It contains all necessary fields as planned (`studentId`, `isAnonymous`, `content`, `rating`, `category`, `status`, `createdAt`).
- **Embedded Data:** The model's key feature is the inclusion of `private List<Comment> thread;` and `private ResolutionLog resolutionLog;`. This design simplifies data retrieval immensely, as all information related to a single

feedback ticket (the original post, all replies, and the final resolution) is stored in one document.

3.2. Repository (`repository` package)

- **`FeedbackRepository.java`:** An interface extending `MongoRepository<Feedback, String>` was created. This provides all standard CRUD methods for the `feedback` collection. A custom method, `List<Feedback> findByStudentId(String studentId)`, was also defined for use in Module 3.

3.3. DTO & Validation (`dto` package)

- **`FeedbackRequestDTO.java`:** This Data Transfer Object was created to define the precise structure of the incoming JSON from the React form.
- **Server-Side Validation:** The DTO utilizes `jakarta.validation.constraints` (e.g., `@NotBlank`, `@NotNull`, `@Min`, `@Max`) to ensure all submitted data is valid *before* it reaches the service layer. This prevents bad data from ever entering the database.

3.4. Service Layer (`service` package)

- **`FeedbackService.java`:** A new `submitFeedback` method was created. This method contains the core business logic:
    1. It receives the `FeedbackRequestDTO` and the `studentId` (from the controller).
    2. It uses the `Feedback.builder()` pattern to construct a new `Feedback` object.
    3. It sets the initial default values as planned: `status` is set to `"open"` and `thread` is initialized as a new `ArrayList`.
    4. It calls `feedbackRepository.save()` to persist the new document to MongoDB.

3.5. API Endpoint (`controller` package)

- **`FeedbackController.java`:** A new controller was created with a primary endpoint:
    - `POST /api/feedback/submit`: This endpoint is protected by Spring Security (requiring an authenticated user).
    - It uses the `Authentication` object (injected by Spring) to get the logged-in user's email.
    - It finds the `User` in the `UserRepository` to get their MongoDB `_id`.
    - It passes the `FeedbackRequestDTO` and the `user.getId()` to the `FeedbackService` to complete the operation.

4. Frontend Implementation (React)

The frontend was updated to provide the UI for submitting feedback.

4.1. Service Layer (`services` package)

- **`FeedbackService.js`:** A new service file was created to manage all API calls to the `/api/feedback` endpoints.
    - `authHeader()`: A helper function was defined to read the user's JWT from `localStorage` and create the `Authorization: Bearer <token>` header.
    - `submitFeedback(feedbackData)`: This function sends the form data via an **Axios** `POST` request to the `/api/feedback/submit` endpoint, including the `authHeader()` to authenticate the request.

4.2. UI Components (`components` package)

- **`FeedbackForm.js`:** This is the main UI component for the module.
    - It uses `useState` hooks to manage the state of each form input: `content`, `rating`, `category`, and `isAnonymous`.
    - A CSS-based star rating system was implemented using radio buttons for accessibility and ease of use.
    - The `handleSubmit` function performs client-side validation, bundles the state into a `feedbackData` object (matching the DTO), and calls `FeedbackService.submitFeedback()`.
    - It provides clear success and error messages to the user.

4.3. Routing and Integration

- **`StudentDashboard.js`:** This component was updated with a `<Link>` from `react-router-dom`, styled as a button, to navigate the user to the new feedback form.
- **`App.js`:** The main router was updated to include the new route:
    - `<Route path="/submit-feedback" ... />`: This route renders the `FeedbackForm` component.
    - Crucially, this new route is wrapped in the `ProtectedRoute` component from Module 1, ensuring that only authenticated students (`role="ROLE_STUDENT"`) can access the form page.