Dynamic Feedback Management System

FEEDBLOOM

Project Repository: https://github.com/sams52s/Dynamic_Feedback_MS

Project Tracking: https://github.com/users/sams52s/projects/2

Assigned Date: Thu, Mar 6, 4:08 PM

Submitted Date: Mon, Mar 24

Requirement Analysis

1. Project Overview

A **Dynamic Feedback Management System** to centralize and streamline feedback collection, tracking, and resolution for in-house products. Currently, feedback is reported informally via Google Chat, leading to inefficiencies such as lost reports, lack of tracking, and delayed resolutions.

The system will be developed using **Java 17** and **Spring Boot** and will include features for structured feedback submission, approval workflows, tracking, analytics, smart filtering, and notifications.

2. Key Functional Requirements

2.1 Feedback Submission

Description:

Users should be able to submit feedback through a structured form.

Features:

- **Fields:**
    - **Title:** Short description of the feedback.
    - **Description:** Detailed explanation of the issue or request.
    - **Category:**
        - **Bug:** Issues or errors in the product.
        - **Feature Request:** Suggestions for new features or improvements.
        - **General:** Other feedback that doesn't fit into the above categories.
    - **Priority:**
        - **Low:** Minor issues or non-urgent requests.
        - **Medium:** Important but not critical.
        - **High:** Critical issues requiring immediate attention.
    - **Attachments:** Users can attach files or screenshots (optional).

Validation:

- Ensure all required fields are filled.
- Validate file types and sizes for attachments.

Technical Details:

- Store feedback in a database with fields for title, description, category, priority, submitter details, and timestamp.
- Use **DTOs (Data Transfer Objects)** to transfer data between layers.

2.2 Feedback Tracking & History

Description:

Each feedback item should have a timeline view showing its history.

Features:

- **Timeline View:**
    - **Status Changes:** Track changes in feedback status (e.g., "Submitted," "Under Review," "Resolved").
    - **Comments:** Additional notes or updates from the team.
    - **Priority Changes:** Any updates to the priority level.
    - **Changed By:** User who made the change.
    - **Timestamp:** Date and time of the change.

Technical Details:

- Create a **StatusChange** entity to log changes.
- Use a **one-to-many** relationship between **Feedback** and **StatusChange** for better tracking.

2.3 Approval Workflow

Description:

Feedback should go through an approval process before being marked as resolved.

Features:

- **Workflow Steps:**
    - **Submission:** Feedback is submitted by a user.
    - **Assignment:** Feedback is assigned to an approver (e.g., team lead or manager).
    - **Review:** The approver either:
        - Approves it for resolution.
        - Requests more information or modifications.
    - **Resolution:** Once approved, feedback is marked as resolved.
- **Role-Based Access:**
    - **Users:** Can submit feedback and view their submissions.
    - **Approvers:** Can update feedback status, priority, and comments.
    - **Admins:** Can manage all feedback and user roles.

Technical Details:

- Use a **state machine** to manage feedback status (e.g., "Pending Approval," "Approved," "Resolved").
- Notify users and approvers via **email** or **in-app notifications** at each stage.

2.4 Smart Filtering & Search

Description:

Users should be able to filter and search feedback efficiently.

Features:

- **Filter Options:**
  - **Category:** Bug, Feature Request, General.
  - **Priority:** Low, Medium, High.
  - **Status:** Submitted, Under Review, Resolved, etc.
  - **Keywords:** Search for specific terms in the feedback title or description.
  - **Date Range:** Filter feedback by submission date.
- **Sorting:**
  - Sort by **submission date, priority, or status**.

Technical Details:

- Implement a **search and filter API** in the backend.
- Use **Pageable** in **Spring Data JPA** for pagination and sorting.

2.5 Analytics Dashboard

Description:

Provide an analytics dashboard to visualize feedback data.

Features:

- **Metrics:**
  - **Most Common Feedback Topics:** Identify recurring issues or popular feature requests.
  - **Trends:** Highlight trends over time (e.g., "Most Requested Feature in the Last Month").
  - **Resolution Time:** Average time taken to resolve feedback.
  - **Feedback Distribution:** Breakdown by category, priority, and status.
- **Visualizations:**
  - Use **charts and graphs** (e.g., bar charts, pie charts, line graphs).

2.6 Notifications & Alerts

Description:

Users and approvers should receive notifications for important events.

Features:

- **Triggers:**
  - New feedback is submitted.
  - Feedback status changes.
  - Comments are added to feedback.
  - Approval is requested.
- **Notification Methods:**
  - Email notifications.
  - In-app notifications.

Technical Details:

- Use **Spring Boot Mail** for email notifications.
- Implement **WebSocket** or **AJAX polling** for in-app notifications.

3. Non-Functional Requirements

- **Performance:** Handle a large number of feedback submissions efficiently.
- **Security:**
  - **Authentication:** JWT-based authentication.
  - **Authorization:** Role-based access control.
  - **Rate Limiting:** Prevent abuse by limiting API calls per user.
  - **Data Encryption:** Encrypt sensitive data in transit and at rest.
- **Scalability:** The system should accommodate future growth.
- **Usability:** Intuitive and responsive user interface.
- **Maintainability:** Well-documented code following best practices.

6. Timeline

Day	Task
1-2	Requirement finalization & system design
3-6	Backend development (feedback submission, history, approval workflow)
7-9	Frontend development (UI for submission, history, filtering)
10-12	Analytics dashboard development

13-14	Security implementation (authentication, authorization)
15-16	Testing (unit, integration, user testing)
17-18	Deployment (Docker)
19-20	Final refinements and documentation

7. Conclusion

The **Dynamic Feedback Management System** will centralize and streamline the feedback process, replacing the inefficient Google Chat-based reporting. By implementing structured tracking, approvals, and analytics, the client can ensure better issue resolution, improved transparency, and valuable insights for future improvements.

User Work Flow

1. User Registration & Authentication

1.1 New User Registration

1. Sign-Up Process:
    - Users access the registration page and provide:
        - Email: A valid email address.
        - Password: A strong, secure password.
    - The system validates the email format and password strength.
2. Email Verification (Optional):
    - A verification email is sent to the user.
    - The user activates their account by clicking the verification link.
3. Role Assignment:
    - The Admin assigns roles: User, Approver, or Admin.
    - The user is notified of their role via email.

1.2 User Login

1. Access & Authentication:
    - Users enter their email and password on the login page.
    - The system validates credentials and generates a JWT (JSON Web Token).
    - The token is securely stored (e.g., in local storage or cookies).
2. Role-Based Redirects:
    - Users are redirected to their respective dashboards:
        - Users: Feedback submission & tracking.
        - Approvers: Feedback review & approval.
        - Admins: User management, analytics, & system oversight.

2. Feedback Submission (User)

2.1 Submitting Feedback

1. Navigation:
    - Users access the Submit Feedback section.
2. Feedback Form:
    - Users provide the following details:
        - Title: Brief description.
        - Description: Detailed explanation.
        - Category: Bug, Feature Request, or General.
        - Priority: Low, Medium, or High.
        - Attachments: Optional file uploads.
    - The system validates input and saves feedback.

    - Feedback status is set to "Submitted".
    - A confirmation message is displayed.
3. Notifications:
    - An email and in-app notification are sent to the assigned Approver.

3. Feedback Tracking & History (User, Approver, Admin)

3.1 Viewing Feedback

1. User View:
    - Users navigate to "My Feedback".
    - Feedback can be filtered by:
        - Category: Bug, Feature Request, General.
        - Priority: Low, Medium, High.
        - Status: Submitted, Under Review, Resolved, etc.
        - Keywords: Search for specific terms.
    - Sorting options: Submission date, priority, or status.
2. Approver/Admin View:
    - Access all feedback via "All Feedback".
    - Advanced filters: Date range, assigned approver.

3.2 Viewing Feedback Details

1. Timeline View:
    - Users click feedback items to see:
        - Status Changes: Submitted, Under Review, Resolved.
        - Comments: Notes from approvers/team members.
        - Priority Changes: Updates to the priority level.
        - Modified By: User who made the change.
        - Timestamp: Date and time of the change.
2. Attachments:
    - Users can view or download attachments uploaded by the user.

4. Feedback Approval Workflow (Approver, Admin)

4.1 Reviewing Feedback

1. Notification:
    - Approvers/Admins receive email and in-app notifications for new feedback.
2. Review Actions:
    - Request More Info:
        - Adds a comment requesting clarification.
        - Status remains "Pending Approval".
        - The user is notified.
    - Approve for Resolution:

        - Assigns feedback to a team member.
        - Status changes to "Under Review".
        - Priority may be updated.
        - The user is notified.
    - Reject:
        - Marks feedback as "Rejected" with a reason.
        - The user is notified.

5. Feedback Resolution (Admin, Assigned Team Member)

5.1 Resolving Feedback

1. Workflow:
    - The assigned team member updates feedback progress.
    - Adds comments to provide updates.
2. Mark as Resolved:
    - Once resolved, the team member changes the status to "Resolved".
    - The system notifies the user.

6. Smart Filtering & Search (All Users)

6.1 Search & Filter Options

1. Searchable Fields:
    - Title: Keywords in the title.
    - Description: Keywords in the description.
    - Category: Bug, Feature Request, General.
    - Priority: Low, Medium, High.
    - Status: Submitted, Under Review, Resolved, etc.
    - Date Range: Filter by submission date.
2. Sorting & Pagination:
    - Results are sortable by submission date, priority, or status.
    - Pagination is implemented for large datasets.

7. Analytics Dashboard (Admin)

7.1 Viewing Analytics

1. Metrics:
    - The admin views:
        - Most Common Feedback Topics: Recurring issues or popular feature requests.
        - Trending Requests: Most requested features over time.
        - Average Resolution Time: Time taken to resolve feedback.
        - Feedback Distribution: Breakdown by category, priority, and status.

2. Visualizations:
    - Charts and graphs (e.g., bar charts, pie charts, line graphs) are used for data visualization.

8. Notifications & Alerts (User, Approver, Admin)

8.1 Triggered Notifications

1. Events:
    - Notifications are triggered for:
        - New feedback submission.
        - Feedback status changes.
        - Comments added to feedback.
        - Approval requests.
2. Delivery Methods:
    - Email Notifications: Sent via Spring Boot Mail.
    - In-App Notifications: Displayed using WebSocket or AJAX polling.

9. Security & Access Control

9.1 Authentication

1. JWT-Based Login:
    - Users log in using JWT-based authentication.
    - Tokens are securely stored and validated.

9.2 Authorization

1. Role-Based Access:
    - Users: Submit and track feedback.
    - Approvers: Review and approve/reject feedback.
    - Admins: Manage users and feedback.

10. Deployment & Maintenance

10.1 Deployment Strategy

1. Containerization:
    - The application is Dockerized for easy deployment.

10.2 Maintenance & Monitoring

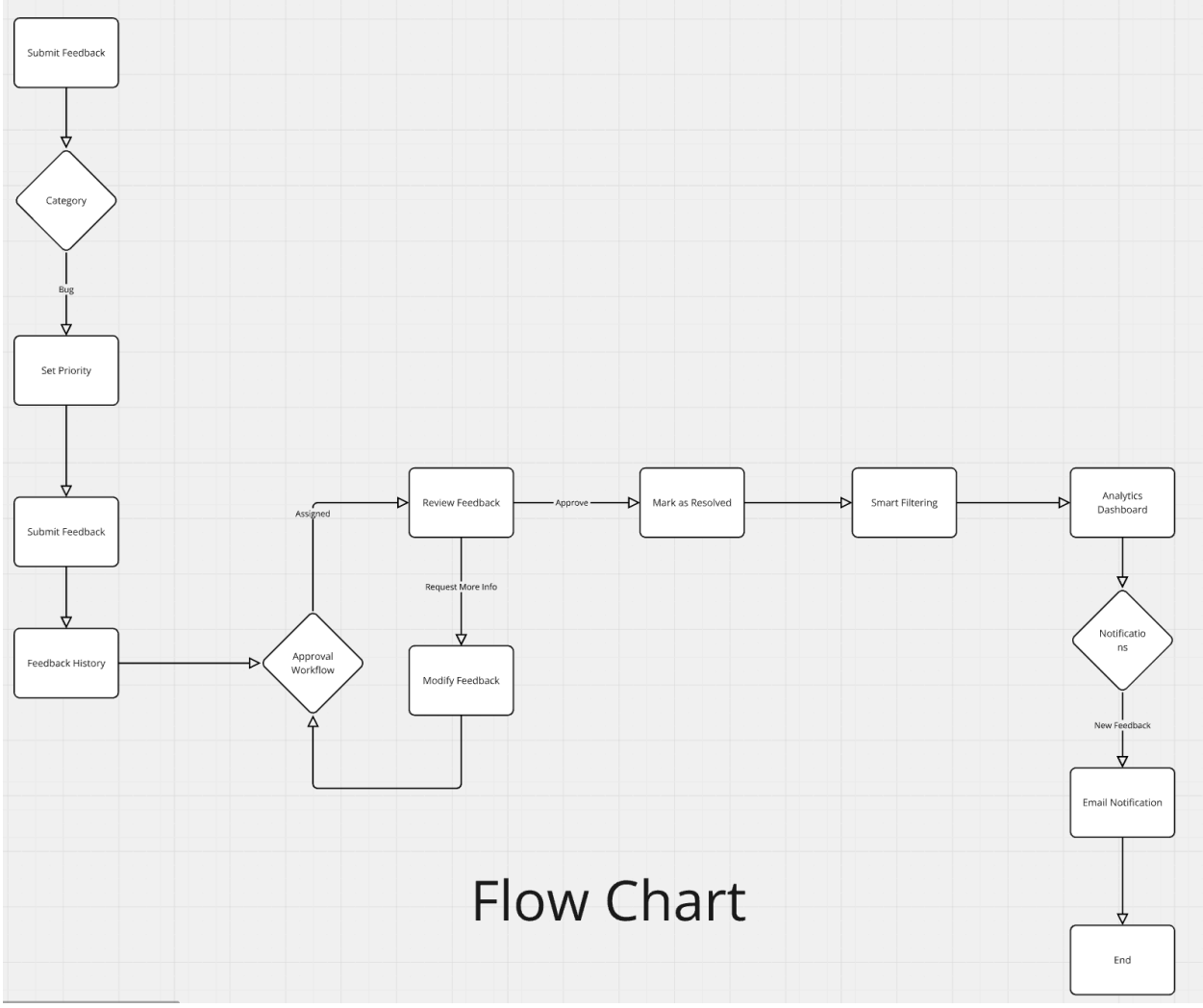
1. Regular Updates:
    - The system is regularly updated with new features and bug fixes.
2. Performance Monitoring:

- Performance is monitored using tools like Prometheus and Grafana.

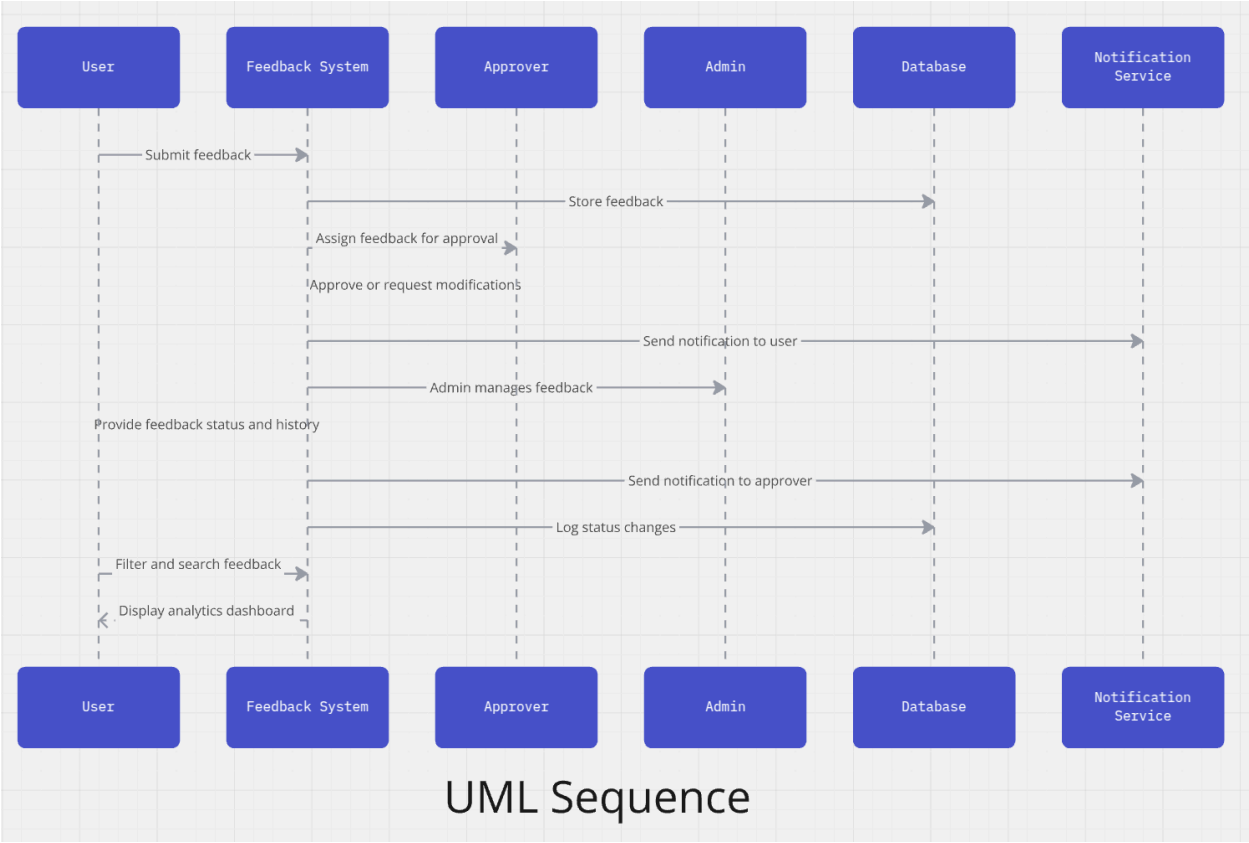
11. End-to-End Flow Summary

1. User submits feedback.
2. Approver reviews feedback.
3. Feedback is approved/rejected.
4. Resolved feedback is tracked.
5. Analytics provide insights.
6. Notifications keep users informed.
7. Admins manage security & access control.

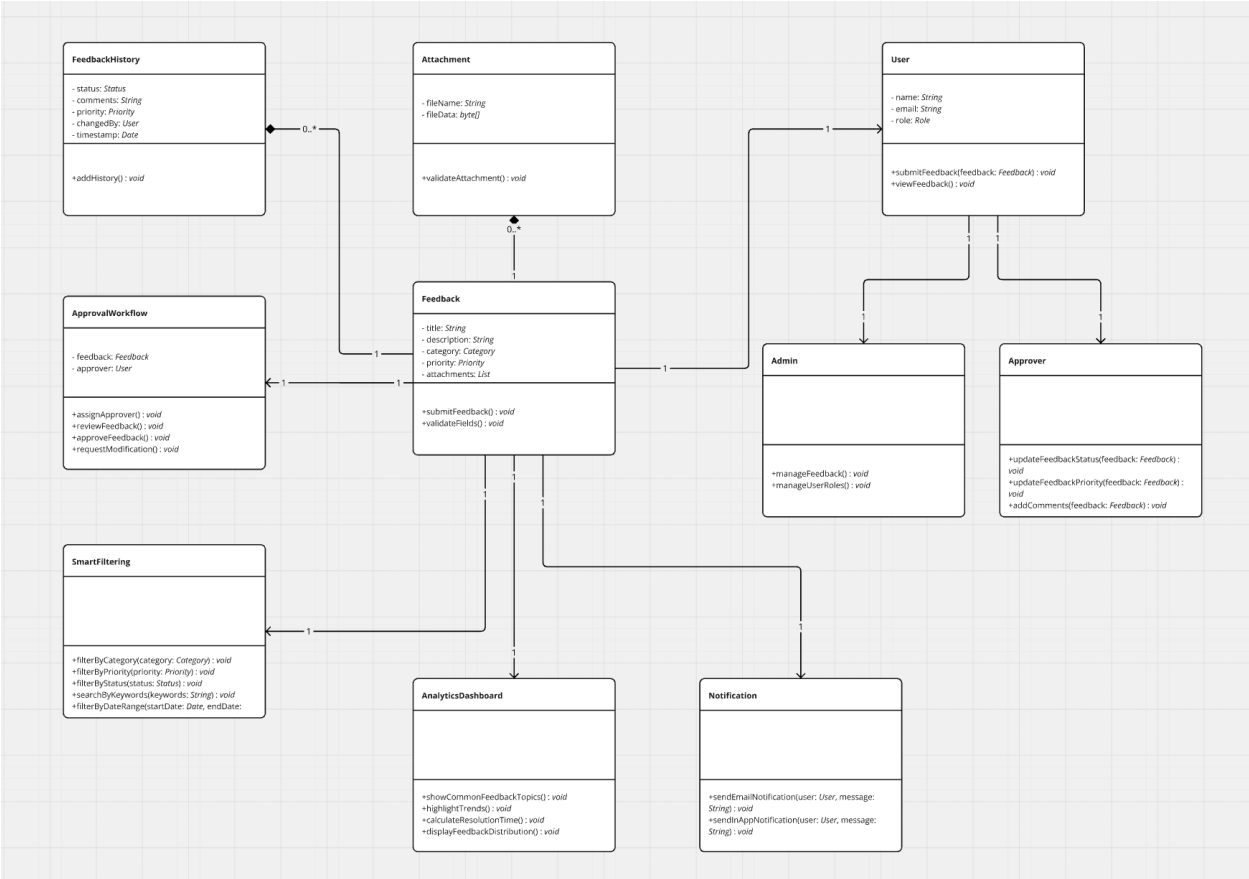
12. FLOW CHART:



13. UML Sequence Diagram:



14. UML Class Diagram:



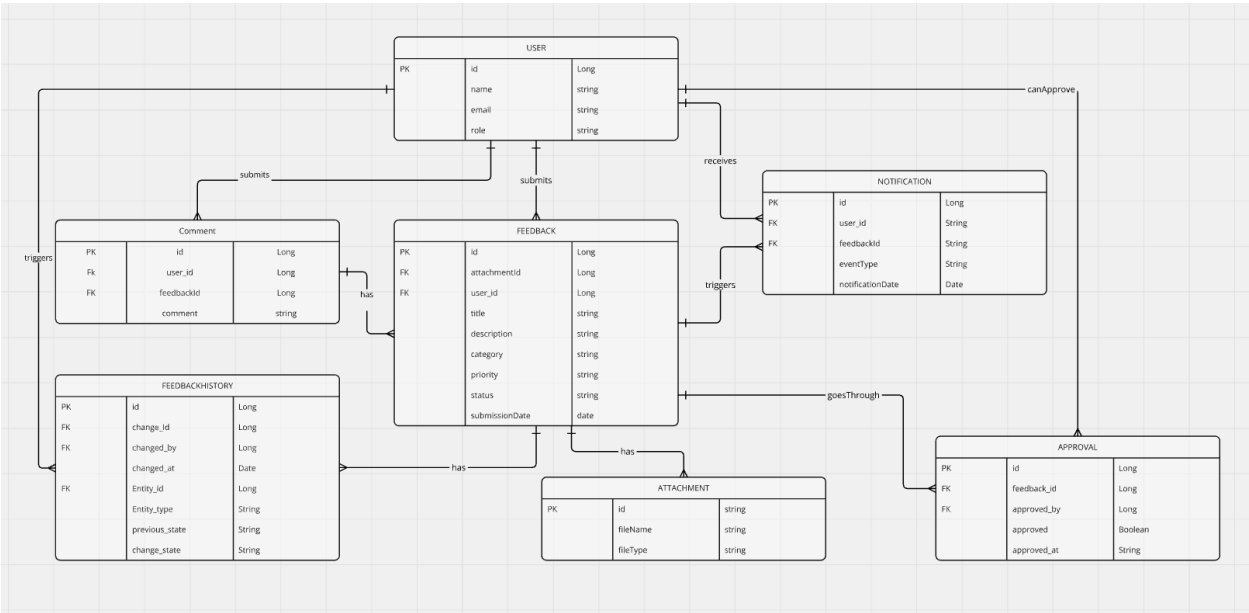
15. Database Schema Design:

Entities & Relationships

1. **User** (`users`)
    - Stores user details and authentication credentials.
    - Roles: `User`, `Approver`, `Admin`.
    - Relationships:
        - A **User** can submit multiple **Feedbacks**.
        - An **Admin** assigns roles to users.
2. **Feedback** (`feedbacks`)
    - Captures user-submitted feedback with categories, priority, and status.
    - Relationships:
        - A **User** submits **Feedback**.

        - An **Approver** reviews **Feedback**.
        - A **Team Member** resolves **Feedback**.
3. **Feedback History** (`feedback_history`)
    - Tracks feedback status changes (audit log).
    - Stores timestamps and the **User** who made changes.
4. **Comments** (`feedback_comments`)
    - Stores discussions and additional info on feedback.
    - Related to `feedback_id` and `user_id`.
5. **Attachments** (`feedback_attachments`)
    - Stores uploaded files related to feedback.
6. **Notifications** (`notifications`)
    - Manages in-app and email notifications for users.
7. **Roles & Permissions** (`roles`, `user_roles`)
    - Implements **role-based access control (RBAC)**.
8. **Analytics & Reports** (`feedback_stats`)
    - Stores aggregated feedback data for quick dashboard queries.

16. ER Diagram Representation:



Submitted Parts

1. User Registration & Authentication

For this feature, I have implemented JWT Authentication along with Spring Security. Although JWT authentication was not mandatory for this project, I incorporated it as an additional enhancement. This ensures that if API-based login or authentication is needed in the future, JWT can be leveraged as a secure and scalable solution.

I have used Thymeleaf conditionals to integrate login and registration forms within the same page, eliminating the need for separate HTML files. This makes modifications easier and enhances maintainability.

2. Security & Access Control

I have implemented Spring Security with Thymeleaf to enforce access control. This ensures that:

- Users can only view their own feedback.
- Managers can access all feedback.
- Feedback comments can only be modified by the original author.

3. Feedback Submission

Users can submit feedback through a form and edit it when necessary. I utilized Thymeleaf templates and fragments to maintain clean and readable HTML code. AJAX is used for updating the form dynamically, preventing unnecessary page reloads.

- Users can update their feedback, and Managers can review and approve it.
- A commenting feature allows users to provide additional input.
- Admins and Managers can also comment if further clarification is needed.
- Users can edit or delete their own comments when required.

4. Feedback Tracking & History

Implemented a tracking system that logs every modification made to a feedback entry.

5. Feedback Approval Workflow

Managers and Admins have the ability to update feedback statuses as part of the approval process.

6. Smart Filtering & Search

Integrated a data table to facilitate advanced searching capabilities. Users can perform various types of searches efficiently. Additionally, some predefined filtering options are available, though a few more need to be added for enhanced usability.

Not Yet Implemented Parts

1. Feedback Resolution

   The assignment functionality is yet to be implemented. This feature will allow Managers or designated individuals to assign feedback resolution tasks to specific users.

2. Analytics Dashboard

   Currently not implemented. This will provide insights and metrics related to feedback submissions and resolutions.

3. Notifications & Alerts

   This feature is yet to be developed. It will notify users about feedback status updates and important actions.