Employee Feedback System

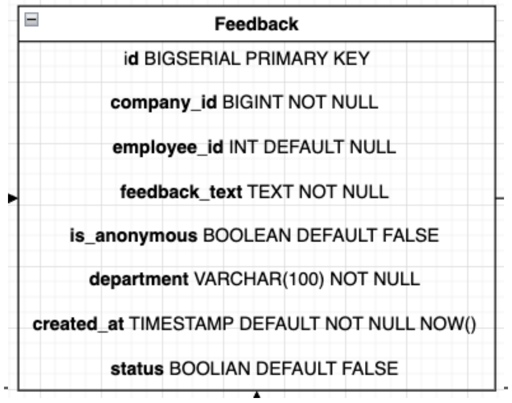
Background

- The Simple Employee Feedback System is a backend service that allows employees to provide feedback on their work experience, either anonymously or identified, which can then be reviewed and responded to by HR or managers. The goal is to foster transparency, improve employee satisfaction, and streamline feedback management within the organization.

Solution Overview

Part 1- Migration Layer:

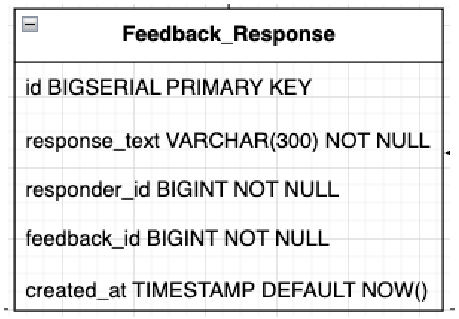
- ○ Purpose: Manages the database schema, creating and updating tables using migration scripts.
  - ☑ ~~**Phase 1- Step 1** Create company table, employees table(Using Eli two postgres db tables)~~
  - ☑ ~~**Step 1.1** Create **feedback table** (employee_id can be null)~~



Feedback
id BIGSERIAL PRIMARY KEY
company_id BIGINT NOT NULL
employee_id INT DEFAULT NULL
feedback_text TEXT NOT NULL
is_anonymous BOOLEAN DEFAULT FALSE
department VARCHAR(100) NOT NULL
created_at TIMESTAMP DEFAULT NOT NULL NOW()
status BOOLIAN DEFAULT FALSE

☑ ~~CREATE INDEX idx_feedback_company_id ON feedback (company_id);~~

- ☑ ~~**Phase 2- Step 1** Create **feedback_response table**~~



Feedback_Response
id BIGSERIAL PRIMARY KEY
response_text VARCHAR(300) NOT NULL
responder_id BIGINT NOT NULL
feedback_id BIGINT NOT NULL
created_at TIMESTAMP DEFAULT NOW()

☑ ~~CREATE INDEX idx_feedback_respose_employee_id ON feedback (feedback_id);~~

☑ ~~CREATE INDEX idx_feedback_response_responder_id ON feedback (responder_id);~~

Tips:

- Indexes data for efficient querying.
- Ensures schema changes do not conflict with existing setups using IF NOT EXISTS.

- Migration version format: VYYYYMMDDHHMM__MigrationName.sql

Part 2- DAO Layer (Data Access Object):

- Purpose: Provides a direct interface to the database, managing operations with jOOQ.
    - ✓ ~~**Step 2.1** Create the tables.kt file in the DAO folder to define table structures using JooqTable. This file acts as a bridge between the database schema and the application, ensuring type safety and consistent access to table fields.~~
    - ✓ ~~**Step 2.2** Create the models.kt file to define data classes that represent the system's entities. These models match the business requirements and database schema, allowing for smooth data handling across layers.~~
    - ✓ ~~**Step 2.3** Create the FeedbackDao file to manage CRUD operations for feedback, followed by creating the FeedbackDaoTest file to validate these methods.~~
    - ✓ ~~**Phase 2-Step 2** Create the FeedbackResponseDao file to handle CRUD operations for feedback responses, and the FeedbackResponseDaoTest file to ensure proper functionality.~~

Part 3- Service Layer:

- Purpose: Implements business logic, validates inputs, and manages exceptions.
- ✓ ~~**Step 3.1** Create FeedbackService.kt file inside the service folder of my project,~~
    - ✓ ~~Functions: insertNewFeedback(newFeedback),~~
    - ✓ ~~getEmployeeFeedbacksHistory(companyId, employeeId),~~
    - ✓ ~~getAllFeedbacks(companyId),~~
    - ✓ ~~**Phase 2- Step 2.1:**~~
    - ✓ ~~getFeedbacksUsingFilter(companyId: Long, createdAt: LocalDateTime?, department: String?,isAnonymous: Boolean?)~~
    - ✓ ~~getFeedbackStatus(searchedFeedback: SearchedFeedback),~~
    - ✓ ~~updateFeedbackStatus(updateFeedback: UpdateFeedbackStatus)~~
    - ✓ ~~**Phase 1- Step 3.1.1** Create a test file for FeedbackService.~~
- ✓ ~~**Phase 2-Step 3** Create FeedbackResponseService.kt also file function: add response to feedback~~
    - ✓ ~~**Step 3.1** Create a test file for FeedbackResponseService.~~

Part 4- Resource Layer:

- Purpose: Exposes REST API endpoints for feedback submission, retrieval, and response.

☑ ~~**Phase 1- Step 4.1** Update AuthenticationResourse.kt file~~

☑ ~~Save in the cookie: Employee role, companyId, employeeId~~

URL	Method	Input	Output	Authorization	Input Format
☑ ~~/api/auth/employee/login~~	POST - Login as an employee(Create a cookie)	{ "firstName":"Rachel", "lastName": "Green", "companyId": 1 }	Response: { OK } Or an exception: 400 Bad Request: Missing or invalid login details. Or 401 Unauthorized: Incorrect credentials.	JWT (Employee/HR/Admin Role)	JSON (Body)
☑ ~~/api/auth/employee/logout~~	POST - Logout + remove cookie	None- removing cookie	Response: { OK } Or an exception: 401 Unauthorized: Invalid or expired JWT token.	JWT (Employee/HR/Admin Role)	None

☑ ~~**Step 4.2** Create FeedbackResource.kt file inside the resource folder of my project.~~

URL	Method	Input	Output	Authorization	Input Format
☑ ~~/api/feedback/submit~~	POST - insert new feedback	Using data class- FeedbackSubmission: { “feedbackText”:”Great work”, “isAnonymous”: false , “department”: “Dev” }	Response: { OK } Or an exception: 400 Bad Request: Missing fields or invalid input data. Or 401 Unauthorized: Invalid JWT	JWT (Employee/HR/Admin Role)	JSON (Body) + cookie
☐ /api/feedback/update/company/{companyId}/feedback/{feedbackId}	PUT - update my feedback Note: that this is not in the spec!	Using data class- FeedbackUpdateRequest: { “feedbackText”:”Bad work”, “isAnonymous”: true }	Response: { OK } Or an exception: 400 Bad Request: Missing feedback text. Or 401 Unauthorized: No permission to update. Or 404 Not Found: Feedback not found.	JWT (HR/Admin Role)	JSON (Body) + cookie
☑ ~~/api/feedback/history~~	GET - get employee feedback history	None	List<Feedback> Or 401 Unauthorized: No access to view history	JWT (Employee/HR/Admin Role)	cookie

☑ ~~/api/feedback/view/all~~	GET - get all feedbacks	None	List<Feedback> Or 401 Unauthorized: No access to get all feedbacks	JWT (HR/Admin Role)	cookie

☑ ~~/api/feedback/view/filter~~	Phase 2-
	POST - get feedbacks using filter
		{
		"createdAt": null,
		"department": null,
		"isAnonymous": null
		}
			List<Feedback>
			Or
			401 Unauthorized: No access to get feedbacks using filter
				JWT (HR/Admin Role)	JSON (Body)
					+
					cookie

☑ ~~/api/feedback/view/status/{feedbackId}~~	Phase 2- GET - get feedback status	None	Status Or 401 Unauthorized: No access to get feedback status	JWT (HR/Admin Role)	Path Params + cookie

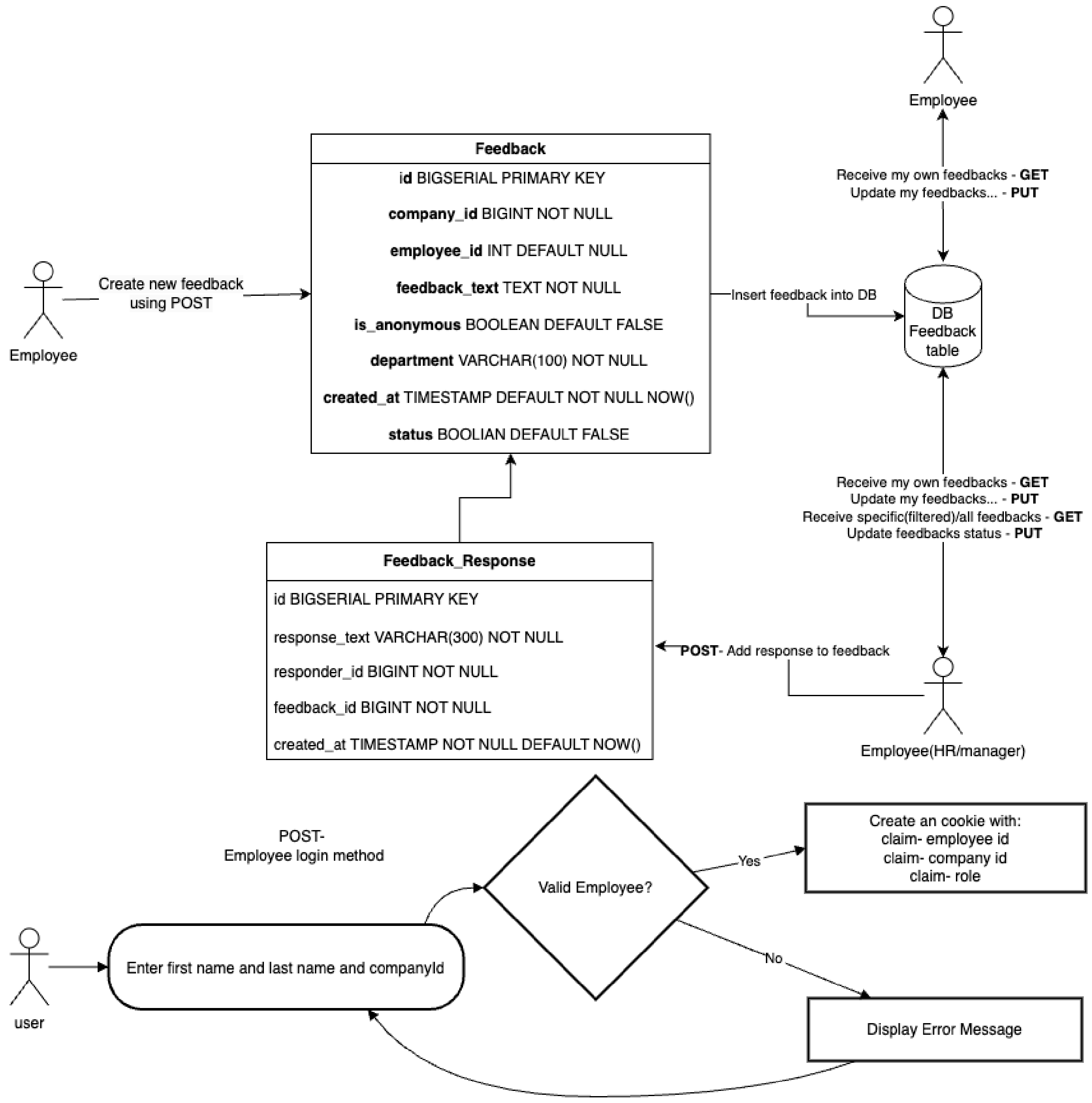
☑ ~~/api/feedback/update/status~~	Phase 2-
	PUT - update feedback status
		{
		"companyId": 1,
		"feedbackId": 99,
		"status": false
		}
			Response:
			{
			OK
			}
			Or
			an exception:
			401 Unauthorized: No permission to update status.
			Or 404 Not Found: Feedback not found.
				JWT (HR/Admin Role)	JSON (Body)
					+
					cookie

☑ ~~**Phase 2-Step 4**(Implement HR Response API) Create FeedbackResponseResource.kt file~~

URL	**Method**	**Input**	**Output**	**Authorization**	**Input Format**

☑ ~~/api/response/submit~~	POST - add response to feedback	{
		
		"responseText": "Thank you!!!",
		"feedbackId": 99
		}
			Response:
			{
			OK
			}
			Or
			an exception:
			400 Bad Request: Missing response text or responder ID.
			Or 401 Unauthorized: No permission to respond.
			Or 404 Not Found: Feedback
				JWT (HR Role)	JSON (Body)
					+
					cookie

High-Level Flow Diagram Description


Employee
Receive my own feedbacks - GET
Update my feedbacks... - PUT
Feedback
id BIGSERIAL PRIMARY KEY
company_id BIGINT NOT NULL
employee_id INT DEFAULT NULL
feedback_text TEXT NOT NULL
is_anonymous BOOLEAN DEFAULT FALSE
department VARCHAR(100) NOT NULL
created_at TIMESTAMP DEFAULT NOT NULL NOW()
status BOOLIAN DEFAULT FALSE
Employee
Create new feedback using POST
Insert feedback into DB
DB Feedback table
Receive my own feedbacks - GET
Update my feedbacks... - PUT
Receive specific(filtered)/all feedbacks - GET
Update feedbacks status - PUT
Feedback_Response
id BIGSERIAL PRIMARY KEY
response_text VARCHAR(300) NOT NULL
responder_id BIGINT NOT NULL
feedback_id BIGINT NOT NULL
created_at TIMESTAMP NOT NULL DEFAULT NOW()
POST- Add response to feedback
Employee(HR/manager)
POST- Employee login method
Valid Employee?
Yes
Create an cookie with: claim- employee id claim- company id claim- role
No
Display Error Message
user
Enter first name and last name and companyId

Testing Strategy

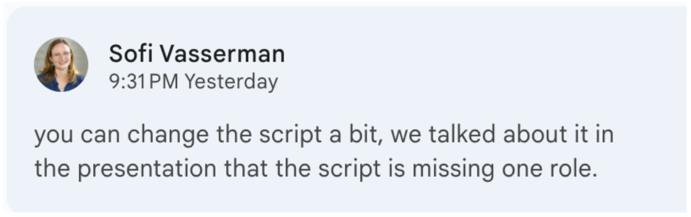
- **Unit Tests:**
  - DAO and Service layers are covered by unit tests to validate business logic and data integrity.
- **Manual Testing:**
  - Resource layer endpoints tested with Postman to ensure proper API behavior and access control.

Security and Validation

- **Authentication:**
  - **JWT tokens** are validated for all incoming requests to enforce proper authorization.

Risks & Problems...

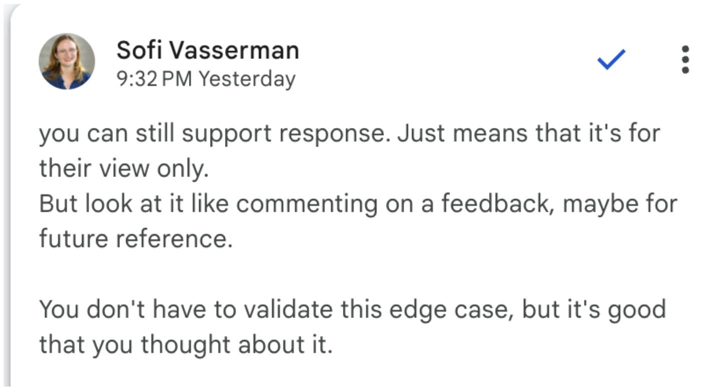
- Check that employee details are not enabled while using anonymous feedback
- I need assistance regarding the role management in our feedback system. Based on the requirements, HR and Admin users should have specific permissions such as viewing feedback, filtering submissions, responding to feedback, and tracking feedback status. However, in the current database schema provided by Eli, the employees table includes roles defined as 'admin', 'manager', and 'employee', but there is no specific role for HR.



Sofi Vasserman
9:31 PM Yesterday

you can change the script a bit, we talked about it in the presentation that the script is missing one role.

- Inability to Respond to Anonymous Feedback: HR or managers are unable to directly respond to anonymous feedback.



Sofi Vasserman
9:32 PM Yesterday

you can still support response. Just means that it's for their view only.
But look at it like commenting on a feedback, maybe for future reference.

You don't have to validate this edge case, but it's good that you thought about it.

Project Schedule for the Feedback System

Phase 1: Basic API Development

Sunday:

Tasks:

- Complete the HLD (High-Level Design) document.
- Obtain final approval from the mentor or project manager.

Monday:

Tasks:

- Perform database migrations (create tables for companies, employees, feedback, and feedback responses).
- Implement necessary improvements based on HLD feedback.
- Start writing the basic DAO files (tables.kt and models.kt).

Tuesday:

Tasks:

- Complete the FeedbackDao implementation and write tests (FeedbackDaoTest).
- Begin work on the Service layer (FeedbackService.kt) focusing on basic feedback functions.

Wednesday:

Tasks:

- Complete the FeedbackService.kt and write tests.
- Finalize the FeedbackResource.kt for basic feedback endpoints (submission, update, and viewing).
- Implement Authentication and Filter (Login, Logout) including saving roles, companyId, and employeeId in cookies.

Phase 2: Advanced Features & Testing

Thursday:

Tasks:

- Implement Phase 2 requirements such as advanced filtering and feedback status updates.
- Develop FeedbackResponseDao and FeedbackResponseService.kt for HR response to feedback.
- Write tests for response functionalities (FeedbackResponseDaoTest and FeedbackResponseServiceTest).
- Finalize the FeedbackResponseResource.kt for HR to respond to feedback.
- Implement endpoints for advanced filtering and updating feedback status.

Project Closing (Friday-Saturday)

Friday:

Tasks:

- Run the application and check all functionalities from Phase 1 and Phase 2.
- Perform minor improvements and bug fixes.

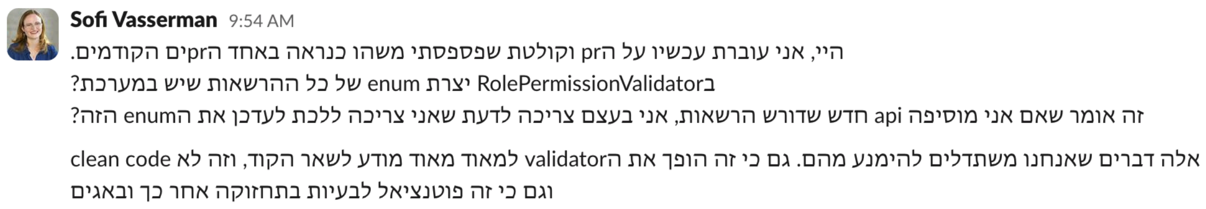
Saturday:

Tasks:

- Finalize the project and prepare it for submission or deployment.
- Add additional server-side functionality if time permits.

הערות להמשך:

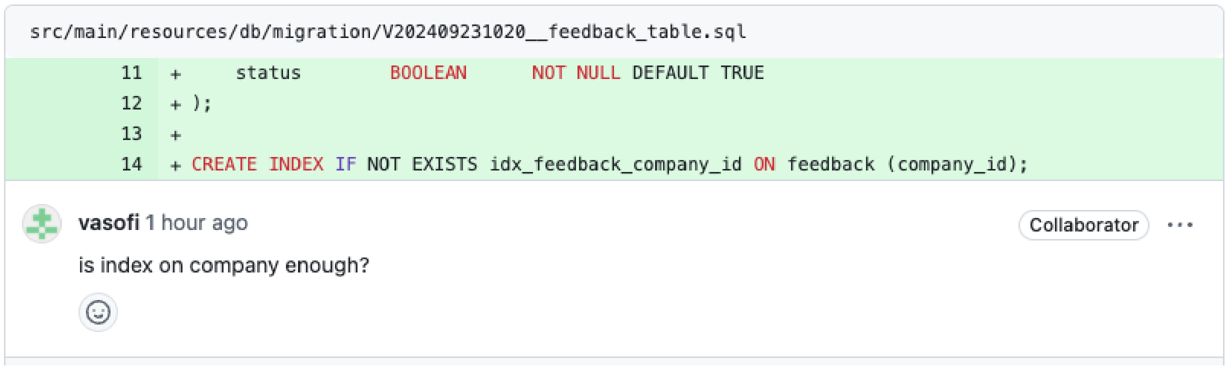
- [x] ~~הערה של סופי:~~


Sofi Vasserman 9:54 AM

היי, אני עוברת עכשיו על הpr וקולטת שפספסתי משהו כנראה באחד הprים הקודמים.
בRolePermissionValidator יצרת enum של כל ההרשאות שיש במערכת?
זה אומר שאם אני מוסיפה api חדש שדורש הרשאות, אני בעצם צריכה לדעת שאני צריכה ללכת לעדכן את הenum הזה?

אלה דברים שאנחנו משתדלים להימנע מהם. גם כי זה הופך את הvalidator למאוד מאוד מודע לשאר הקוד, וזה לא clean code וגם כי זה פוטנציאל לבעיות בתחזוקה אחר כך ובאגים

- [x] ~~לשנות את השגיאה בשליחת response לשגיאה של bad request exception~~
- [x] ~~להוסיף אינדקסים - אינדקסים על שדות שאנחנו שולפים ועושים עליהם תנאים באופן תדיר, כי המטרה של האינדקס זה לייעל את השליפות שלנו. אם אין אינדקס, אז הדיבי סורק את כל הטבלה.~~


src/main/resources/db/migration/V202409231020__feedback_table.sql

```
+     status        BOOLEAN      NOT NULL DEFAULT TRUE
+ );
+
+ CREATE INDEX IF NOT EXISTS idx_feedback_company_id ON feedback (company_id);
```

vasofi 1 hour ago Collaborator ···

is index on company enough?

- [x] ~~Implement validation for feedback content (e.g., enforce a (כנראה התפספס לי)לא בוצע עדיין minimum length of feedback~~

תוספות שלי:

1. יצירת unique index לעובדים, לא יכולים להיות שני עובדים עם אותו שם ואותו שם משפחה באותה חברה