asn2
2026/2/15

24 Points Possible

Attempt 1

In Progress
NEXT UP: Submit Assignment

Add Comment

Unlimited Attempts Allowed

Details

Assignment 2: Full CRUD Spring Boot + PostgreSQL + Thymeleaf + Testing + Design

Summary

Build a full CRUD web application: **“Rate CS Teaching Staff”** using:

- Java + Spring Boot
- Thymeleaf templates
- PostgreSQL database
- A test suite that covers each feature (create/read/update/delete + validation)

The app will allow users to view, add, edit, and delete ratings for teaching staff. This is an initial "iteration 1" version and not a full application. For example, it allows users to see teaching staff ratings, but it does not currently provide the ability to calculate an average rating for a specific instructor. Such a feature might be added in a future update.

Learning Objectives

- Implement a complete MVC CRUD workflow with persistent storage.
- Apply server-side validation (including email validation) with user-friendly error messages.
- Write a meaningful test suite (unit + slice/integration) that guards features.
- Demonstrate OO design and (where appropriate) polymorphism.

Domain Model: `StaffRating`

Required attributes (minimum):

- `id` (auto)
- `name` (required, length constraints)
- `email` (required, must be valid email format; must be unique OR explain your rule)
- `roleType` (enum: `TA`, `PROF`, `INSTRUCTOR`, `STAFF`, etc.)
- `clarity` (integer 1–10)

- `niceness` (integer 1–10)
- `knowledgeableScore` (integer 1–10)
- `comment` (optional, length limit)
- `createdAt`, `updatedAt` (recommended)

Pages (Thymeleaf)

1. **Index page**: list all staff ratings (summary view)
   - shows name, roleType, and an overall score (aggregate or average of *clarity*, *niceness*, and *knowledgeableScore*)
   - links to a detail page for each entry
2. **Detail page**: shows full attributes
   - buttons/links: Edit, Delete
3. **Create page**: form to add a new staff rating
4. **Edit page**: form to edit an existing staff rating
5. **Error/validation display**: all forms must show field-specific error messages

Required Features (CRUD)

Create

- Create a staff rating via form POST
- Reject invalid input with clear error messages (don't crash)

Read

- List view (all entries)
- Detail view (one entry by id)

Update

- Edit an entry by id
- Preserve values if validation fails and redisplay errors

Delete

- Delete by id (with a confirmation step OR a clear UX pattern)

Validation Requirements (Must-Haves)

- Email must be a valid email format (server-side)
- Scores must be integers within [1,10]
- Name must be non-empty
- At least one additional validation rule of your choice (e.g., comment length, unique email, etc.)

Testing Requirements (Write tests for each feature)

You must include tests that cover:

1) Validation tests

- invalid email rejected
- out-of-range score rejected
- missing required fields rejected

2) Web / controller tests (MockMvc or Spring test)

- GET index returns 200 and includes expected model attributes
- POST create:
  - success path redirects appropriately
  - failure path returns form with errors

3) Persistence tests

- saving and retrieving entries works
- delete removes the entry

> You may choose either “slice tests” (@WebMvcTest, @DataJpaTest) or full integration tests (@SpringBootTest). Just be consistent and explain your choice in a short TESTING.md.

OO Design / Polymorphism (Required “Design” Component)

Role-based behaviour

- Define an abstract `StaffMemberProfile` (or interface) with a method `displayTitle()`
- Provide concrete implementations for at least **two** roles (e.g., TA vs PROF)
- Use the polymorphic behaviour somewhere in the UI (detail page or list)

Keep it simple—this is about demonstrating good design, not building a huge hierarchy.

Deployment (Render + PostgreSQL)

- Deploy the app on Render
- Use a Render Postgres database
- Use environment variables for DB credentials (no secrets committed)
- Provide a README section: “How to run locally” and “How to deploy.”

Deliverables (Submit a single text file on Canvas)

1. Render URL
2. GitHub repository URL
3. A short “Known Issues / Future Work” section, which outlines known bugs or features that you didn't get to finish.

Repository must include:

- `README.md`
- `TESTING.md` (what you tested and how to run tests)

Marking

- CRUD correctness + UX clarity: 40%
- Validation + error handling: 15%
- Database + persistence correctness: 15%
- Test suite quality and coverage of required behaviours: 20%
- OO design/polymorphism: 10%

View Rubric

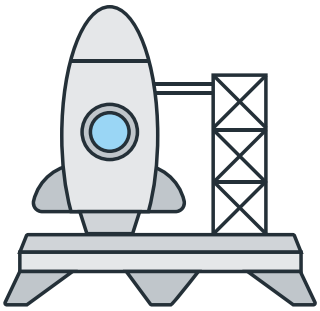
assignment 2

Criteria	Ratings		Pts
Ability to view, add, delete and edit. The code should be documented accordingly.	**8 pts** **Full Marks**	**0 pts** **No Marks**	/ 8 pts
Validations view longer description	**3 pts** **Full Marks**	**0 pts** **No Marks**	/ 3 pts
Implementation Design view longer description	**2 pts** **Full Marks**	**0 pts** **No Marks**	/ 2 pts
Test suite	**4 pts** **Full Marks**	**0 pts** **No Marks**	/ 4 pts
Persistence	**3 pts** **Full Marks**	**0 pts** **No Marks**	/ 3 pts
			Total Points: 0

Choose a submission type

Upload

More



Choose a file to upload

or

Webcam Photo

Canvas Files

Submit Assignment