Spring Boot Project Assignment

Full-Stack Application with DB, Authentication, Sessions, Pagination & Unique Features

Backend: Spring Boot
Frontend: Any stack (Thymeleaf/React/Vue/Angular/HTML/CSS)
Database: SQL or NoSQL

Each team should have 10 members maximum.

1) Goal & Originality

Build a unique, production-style web application that solves a real problem or meaningfully improves a workflow. “Unique” means not just plain CRUD: include all the fully functioned functionalities.

2) Functional Requirements

1. User Accounts & Authentication
   • Sign up, Sign in, Sign out
   • Session & Cookies based login. If you use JWT, also demonstrate a session/cookie flow
   • Role-based authorization with protected routes.
2. Domain Features
   • At least two core entities with relationships.
   • Full CRUD for the main entity.
   • Pagination and sorting or filtering on at least one list screen.
3. Validation & Error Handling
   • Bean Validation on inputs with friendly UI messages.
   • Centralized error handling with proper HTTP status codes for API calls.
4. State & Security
   • Demonstrate sessions and cookies (HttpOnly; set Secure & SameSite in prod profile).
   • Implement CSRF protection for form posts.
5. Data & Persistence
   • Use JPA (SQL) or a NoSQL client (e.g., MongoDB). Include schema/collections & seed data.

3) Technical Requirements (Backend)

- Spring Boot, layered architecture: Controller → Service → Repository → Entity (Model)
- Spring Data JPA (SQL) or appropriate NoSQL driver
- Spring Security for authentication & authorization
- Validation
- Pagination
- Error handling
- API documentation

4) Beyond CRUD — Pick at least two

- Advanced Search/Filtering
- File Uploads (images/docs) with server-side validation
- Email/SMS Notification (verification codes, order confirmations)
- Scheduling for reminders/automation
- Caching (Spring Cache) for hot endpoints
- External API Integration (payments, maps, weather, etc.)
- Soft Deletes / Audit Trail (created/updated/by)
- Real-time Updates (WebSocket/SSE)
- Reporting/Export (CSV/PDF for a list view)

5) Frontend Expectations

- Usable, accessible UI that covers all features
- Handle auth (cookie/session or token). For cookie/session, set credentials true & secure cookie flags.
- Use form binding, validation errors, and method overrides for PUT/DELETE.

6) Non-Functional Requirements

- Performance
- Security
- Reliability: handle 404, 400, 401/403 gracefully
- Code Quality: clear packages, naming, no dead code
- Testing
- Documentation: README with setup, run steps, and screenshots

8) Submission Package

- GitHub repository (public or classroom private) with clear commit history
- README.md
- Postman collection