Design Brief:

1. Application Overview

This document outlines the design of a simple, secure web application for managing a medical register. The system provides CRUD (Create, Read, Update, Delete) functionality for patient records. It is built with a modern Java stack, featuring a Spring Boot 3 backend and a dynamic XHTML frontend, with robust security and a clear path for future expansion.

2. Assumptions

- **Authentication:** The primary assumption is that user authentication is handled externally by Auth0, which is integrated with an existing LDAP directory for user management. The Spring Boot application will not connect to LDAP directly but will delegate authentication to Auth0 via the OAuth2/OIDC protocol.
- **Database:** For development and assessment purposes, an in-memory H2 database is sufficient. A production environment would use a persistent database like PostgreSQL or MySQL.
- **Frontend:** XHTML is rendered server-side using Thymeleaf, which is ideal for its seamless integration with Spring and its ability to produce well-formed markup.
- **Deployment:** The application will be packaged as a standalone JAR file, suitable for deployment in any cloud or on-premises environment with a Java runtime.

3. Integration Considerations

- **Auth0/LDAP:** The integration point is the application.properties file, which holds the Auth0 client credentials. No direct LDAP code is needed in the application. The key consideration is managing secrets (client ID and secret) securely, for example, using environment variables or a secrets management service in production.
- **Frontend-Backend:** The connection is managed by Spring's Model object and Thymeleaf's templating. The controller populates the model with data from the repository, and Thymeleaf renders it in the XHTML view. This creates a tightly coupled but simple-to-develop architecture.

4. Future Considerations

- **API Layer:** Introduce a RESTful API layer (@RestController) separate from the UI-serving @Controller. This would allow the backend to support other clients, such as a mobile application.

- **Database Migration:** For production, migrate from H2 to a more robust database like PostgreSQL. Implement a database migration tool like Flyway or Liquibase to manage schema changes.
- **Advanced Auditing:** Expand the auditing feature to log which user performed which action (e.g., createdBy, lastModifiedBy) by integrating Spring Security's context with the JPA auditing mechanism.
- **Containerization & Scalability:** Package the application in a Docker container and manage it with Kubernetes for better scalability, resilience, and deployment consistency across environments.
- **Enhanced Search:** Implement more advanced search capabilities, such as searching by specific medical conditions or date ranges, potentially using a dedicated search engine like Elasticsearch.