CST-339 CLC Project Overview

Introduction

In this course, you will design and build a complete enterprise-class N-Layer application using the Spring Boot framework. The application must adhere to a specific set of high-level functional and technical requirements, as shown in Figure 1 below. You will have the freedom to pick the application you want to design and develop. You will work in teams of no more than 2, unless there is an odd number of students, in which case a team of 3 will be acceptable. Example applications could include, but are not limited to, an ecommerce application, customer management application, order management application, blog site application, DVD/book/music management application, etc.

Each milestone delivery will include a Design Report that captures the technical approach, design decisions, UI wireframe designs, sitemap designs, ER database designs, and project risks/issues. All code developed during the project will be maintained in a GCU-approved private GIT repository.

You will need approval for the project chosen from your instructor prior to moving onto the detailed design and development of your application.

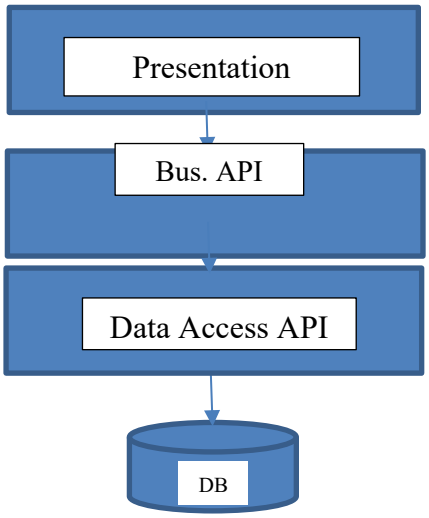

Figure 1 *N-Layer Logical Design*

- Spring MVC (Views, Models, and Controllers)
- Spring Core (Business and Data Services)
- Spring Data JDBC (Persistence)
- Spring Core (IoC and DI)
- Spring REST Services (APIs)
- Bootstrap (Responsive Design)
- Spring Boot Embedded Tomcat (as a JAR)
- MySQL (Relational Database)

The design and code must support the following high-level requirements:

- The application must implement a user registration module and a login module.
- The application must be designed using an N-Layer architecture, with distinct and separate presentation components, business services, and persistence services.
- The application must adhere to industry best practices, exception handling, and error handling as discussed either in topic readings, lectures, or provided as peer code review feedback.

- The application presentation must be entirely written using Spring MVC compliant pages using Thymeleaf templates.
- The application must perform data validation on all form data entry fields.
- The application must support a responsive design using the Bootstrap framework.
- The application must **not** have business logic or business rules implemented in Spring MVC views, models, or controllers.
- The application must implement all CRUD methods on whatever business domain is being addressed (i.e., products, music, blogs, etc.).
    - A page that lists all "products" as a tabular report.
    - A page that allows a user to create a new "product."
    - A page that allows a user to display the details of a "product."
    - A page that allows a user to update an existing "product."
    - A page that allows a user to delete an existing "product."
- The application must use a relational database, such as MySQL or PostgreSQL. It should be noted that if you develop your application using Spring Data JDBC and use the CrudRepository, you could easily port your application over to use MongoDB, but this will not be possible until you finish Topic 5. You must receive permission from your instructor to use MongoDB.
- The application must use the Spring JDBC or Spring Data JDBC to access the database.
- The application must use Spring Beans to implement all business services and persistence services.
- The application must use proper declarative Spring annotations within all components and use dependency injection (DI) for all models, controllers, services, and resources required by the application.
- The application must be deployed using Spring Boot and use the embedded Tomcat (version 9.x or later).
- The application must **not** be able to access secure pages (all but the root, registration, and login pages) without first logging into the application. The application will automatically redirect the user to the login page if they try to access a secure page without first logging in. The application must be secured using Spring Security form-based authentication and a database.
- The APIs must not be anonymous, must be secured using Spring Security, and at a minimum, must use Basic HTTP Authentication using a database.
- The application classes must be fully documented using JavaDoc format.

Project Milestones

The team project will be designed and built using an iterative approach and delivered over the following 8 milestones. It should be noted that all milestones will include a Design Report, except for Milestone 1, and delivery of application code to support the appropriate milestone requirements. It is expected that the code will be refactored during each iteration based on peer code review or instructor feedback.

Milestone 1:

- Project Proposal, Draft Sitemap, and Draft Division of Work across the team

Milestone 2:

- Main Application Module (using Spring MVC)
- Registration Module (using Spring MVC)
    - Without a database.
- Login Module (using Spring MVC)
    - Without a database
- Responsive Design (using the Bootstrap Framework)
- Page Layouts using Thymeleaf Layouts
- Design Report
- Screencast running on local development environment

Milestone 3:

- Product Creation Module (using Spring MVC and Spring Core)
    - Registration Module and Login Module will be refactored to use Spring Beans and IoC.
    - Without a database
- Updated Design Report
- Screencast running on local development environment

Milestone 4:

- Refactoring
    - Registration Module and Login Module will be refactored to use using Spring JDBC or Spring Data JDBC.
    - Product Creation Module will be refactored to use using Spring JDBC or Spring Data JDBC.
- Built as a JAR file using Maven.
- Updated Design Report
- Screencast running on local development environment

Milestone 5:

- Product Display Module (using Spring MVC and using Spring JDBC or Spring Data JDBC)
- Product Update/Delete Modules (using Spring MVC and using Spring JDBC or Spring Data JDBC)
- Updated Design Report
- Screencast running on local development environment

Milestone 6:

- Refactored Login Module (using Spring Security)
    - Use Form-based authentication and a database.
    - All pages, excluding the Login and Registration Modules, will be protected and secured using Spring Security.
- Updated Design Report
- Screencast running on local development environment

Milestone 7:

- REST APIs:
    - REST API design using Microsoft Word template or Swagger
    - REST API 1: Return all products.
    - REST API 2: Return a desired product.
    - All APIs must be secured using Spring Security and, at minimum, use Basic HTTP Authentication using a database.
- Design and Code Cleanup
- Updated Design Report

Milestone 8:

- JavaDoc Generation
- Final Application Code
- Final Project Presentation
- Final Design Report