Final Project Specification

The purpose of your Final Project is to produce a comprehensive multi-layer enterprise application, consisting of a persistence layer, service layer, and presentation layer, also making use of services provided by the container such as security and transactions.

The assignment consists of 1 **Maven Web Application** project, and should build on your prior work using the Payara application server:

- Your domain model as the persistence layer
- Your EJB or Service components as the service layer
- Your security structures and configuration

For the final project, you will be adding **functionality** and **presentation** to your prior work using JSF. You are always free to re-design or re-factor any of your prior work, but if you followed the lab specifications, you should be able to use them as the **foundation** for your final project. You should expect to enhance and add to your previous labs for the Final Project.

Graduate students are expected to complete the entire specification. **Undergraduate** students are encouraged to complete the entire specification, but can omit the "admin requirement" to provide functionality for creating a new user in the system.

Time Management

The Final project has a hard deadline. ***No*** late work can be accepted for the final project. Manage your time carefully. ***Do not wait*** until the last minute to work on your project. You are being given ample time to produce a comprehensive final project, and we expect your output to reflect that. Furthermore, if you have been meeting the due dates of your labs, they have been structured to help you build your final project.

Project Requirements

Start with Lab 10, which includes all cumulative lab work.

All requirements from previous labs apply to the Final Project, as related to the persistence layer, service layer, and security layer.

New requirements for functionality and presentation are as follows:

- Make appropriate use of JSF capabilities for re-usability, for example using **templates** for layout purposes, and **composite components** in your JSF views.
- Follow the separation of concerns principle and make appropriate use of the various application layers. For example, use fields in your JSF Controllers as the "MVC model" and Facelets XHTML pages as your "MVC views."

- **Graduate Students:** provide "admin functionality" to create a new user account in your system. For example, on your login screen, "If you do not have an account, click here to sign-up" (or create an admin role that has access to functionality for creating new users)
- Provide functionality for browsing, finding, and selecting entities based on various criteria. Consider your application roles when designing and implementing this functionality.
- Provide functionality for displaying entities in tabular format where appropriate, or in a single-record form where appropriate. Consider your application roles when designing and implementing this functionality.
- Provide functionality for adding, deleting, and modifying entities. Consider your application roles when designing and implementing this functionality. Not all roles should be able to manipulate all entities.
- Include appropriate navigation between pages, including the ability to return "home." Do not leave dead-end pages with no navigation. Consider usability in all aspects of your design and layout.
- Ensure server-side validation of user input, and display appropriate messages and navigation if user input fails to validate. For JSF, this means appropriate use of Bean Validation and the h:messages tag. Use of client-side validation is fine as well - but is optional. Server-side validation is **required**.
- Basic CSS with Bootstrap is highly encouraged.
- Provide enough sample data to demonstrate your functionality and security model, and to show that your application is usable and testable.
- Make appropriate use of logging and exception handling, such that the user receives messages appropriate for a user, whereas technical messages are logged by Payara.

Your end result should be a fully functional application for your domain model. I should be able to run your project and interact with it based on the information you provide in your README documentation.

I will provide a link to my in-class Example project(s) in Canvas. You are welcome to follow the design and patterns I use in my examples, but if you find yourself doing so please make sure you provide a reference in your javadoc comments and/or README explaining that you based your project on "Instructor Example" or "Example from Web." Always provide references and citations when you use ideas from other sources.

My examples are not provided for you to copy, but to illustrate common patterns and approaches. For example - I do not look favorably on seeing my instructional comments left in code samples - it shows me that you haven't even taken the trouble to apply your own coding style, thoughts, ideas or comments to your projects. Keep this in mind!

If you find it beneficial to use additional libraries or frameworks in your project, you are free to do so provided you follow the general project requirements and fully document their use in your README. I encourage you to explore and be creative. Document any additional libraries used in your README documentation.

Make sure you are using standard Java naming conventions as enforced by our IDEs. These conventions used to be documented by Sun Microsystems but the pages are no longer maintained by Oracle in anything but a legacy format.

If you use code from a tutorial, hint or example, you must appropriately reference the source and give credit to the author. You may use your Javadoc comments to do so.

Documentation

Comment your code appropriately. Use standard javadoc format. **Generate** javadocs and move to your web root. **Link** to your javadocs from your welcome/index page.

Create a page named **Final Project Readme**. Include the following sections. I have included sample questions for you to think about as you write these sections, but you should write in your own words. The questions are meant to give you a guideline, not as a literal outline:

- Project Summary
  - Use this section to describe the project in your own words.
  - How did you fulfill each requirement of the specification?
- Design
  - Use this section to describe the design of your final project's functionality.
  - What functionality did you implement?
  - How does navigation flow from one functional area to another?
  - Also, use this section to list any extra credit you have implemented, and how the additional features were incorporated into your design, including your insights.
- Requirements (Installation, Compile, Runtime, Database, etc)
  - Use this section to explain how I should install, build and run your project.
  - Write it step-by-step as if I do not have any knowledge of how to do so.
  - What are the versions of the tools, libraries and API's used in your project?
- Screen Captures
  - Include enough screen captures to illustrate your working project.
- Expected Results/Known Issues
  - Use this section to describe any known issues with your project. Nothing is ever perfect, and it is better to document issues than ignore them.
  - Also, provide me with a known working test script to follow when I run your project. For example:
    - Login as user1 with password foo
    - Enter a customer name of "Fred" in the search box
    - etc
- Development Insights
  - Use this section to tell me anything you want about the project, and your design/development experience during the project.
  - What did you learn?
  - Was there something you would like to explore further?
  - What did you like, or not like?

Extra Credit (5 points each unless noted – for both UG and GR students)

We may cover some of these in class, but others will require you to research. When you are assigned projects in the workplace, you will often need to research and apply new technologies independently, and it is always important for you to try new things. My hope is that the opportunity for extra points will encourage you to try these advanced Jakarta EE concepts. Wherever possible, I have tried to include items that have examples in our textbook.

- Incorporate an EJB Timer Service in your application
- Send an email using a JavaMail Resource for your application. This should make sense for your domain model. For example, send an email reminder based on an assignment deadline or an email reminder or confirmation for an upcoming appointment. Be creative, but be careful that you don't spam anyone. **DO NOT** use IIT mail servers for this.
- Use a Stateful EJB for some functional part of your application, such as a shopping cart. Be creative!
- Implement a Web Service layer that uses your *existing* service layer to expose either a SOAP or REST API. Demonstrate this with either a test case, stand-alone Java client, shell script, or batch script.
- Implement a Bean Validation Custom Validator
- Incorporate PrimeFaces or another JSF component library into your application to explore the use of additional JSF component libraries beyond what basic JSF provides.
- Incorporate the WebSocket API into your application in a manner of your choosing
- Have your own idea for an extra credit enhancement? Email it to me! If it is approved, it will be worth extra credit.

Submission

Submit a zip file of your project called uidFP.zip and submit to the Canvas or Beacon assignment link. For example, mine would be spyrisonFP.zip.

You must perform an IDE **clean** operation on your project before zipping. I do not need your target folders, and I will mark off points if you submit them.

Late Final Project submissions can not be accepted. There is a deadline for submitting final grades to the registrar, and I have given you as much time as possible.