The purpose of this assignment is to produce a comprehensive multi-tiered 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, using the Glassfish application server:

Your domain model as the persistence layer: *Medical App, which patients can schedule, cancel an appointment, Doctors can view their appointments and lookup patient's info.*

Start by creating three security roles Doctors, Patients and Nurses. Add the following user to the users.

Username: Doctor1
Password: P@ssword1
Username: Patient1
Password: P@ssword1
Username: Nurse1
Password: P@ssword1

Your EJB or Service components as the service layer

Your security structures and configuration

I assuming you will use JSF as your MVC framework, that being said, you are free to use whatever Java MVC framework you prefer, or you can build your own using Servlets and JSP.

Whatever MVC framework you select, makes appropriate use of its capabilities for re-usability, for example using templates (or includes) for layout purposes, and composite components in JSF.

Whatever MVC framework you select, follow the separation of concerns principle and make appropriate use of the various application layers. For example, use Backing Beans as your model and Facelets as your view in JSF.

- Provide 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 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.
- 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/or JSF Validation. Basic CSS 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
- Implement an admin user and role that can manage Users, Groups, and whatever other functionality makes sense for your application. It all depends on your design choices. Be creative!
- Incorporate the 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 an email reminder for an upcoming appointment.
- Implement a JSF Custom Validator OR a Bean Validation Custom Validator
- Implement a JSF Custom Converter
- Implement the Comparable interface on your Entity classes. This enables the use of many default sorting algorithms within the Collections framework. Note, if you have already done this, you can't do it a second time.
  (http://docs.oracle.com/javase/tutorial/collections/interfaces/order.html)

Your end result should be a fully functional application for your domain model.

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.

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.

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?

Requirements (Installation, Compile, Runtime, 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

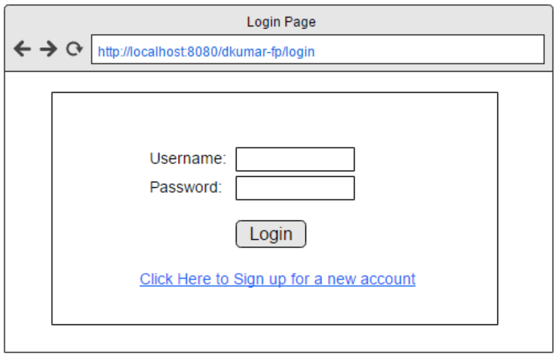
Include enough screen captures to illustrate narrative to prove project is working.

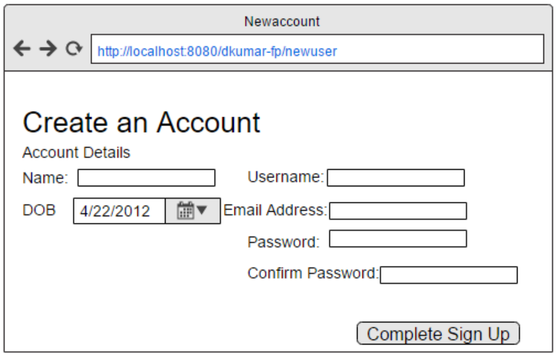
-

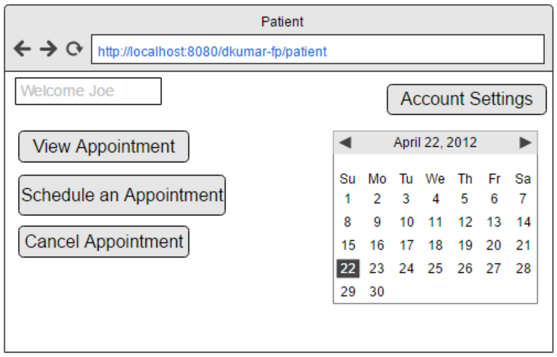
Expected Results/Known Issues

- 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/password

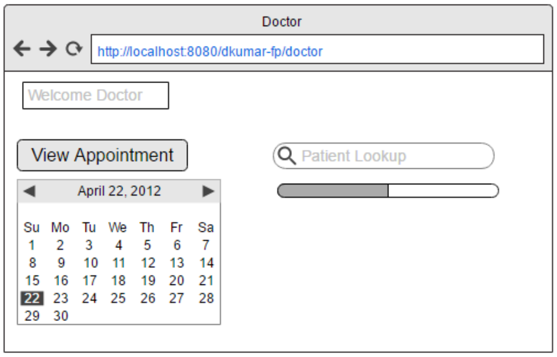
Sample mockup:


Login Page
http://localhost:8080/dkumar-fp/login
Username:
Password:
Login
Click Here to Sign up for a new account

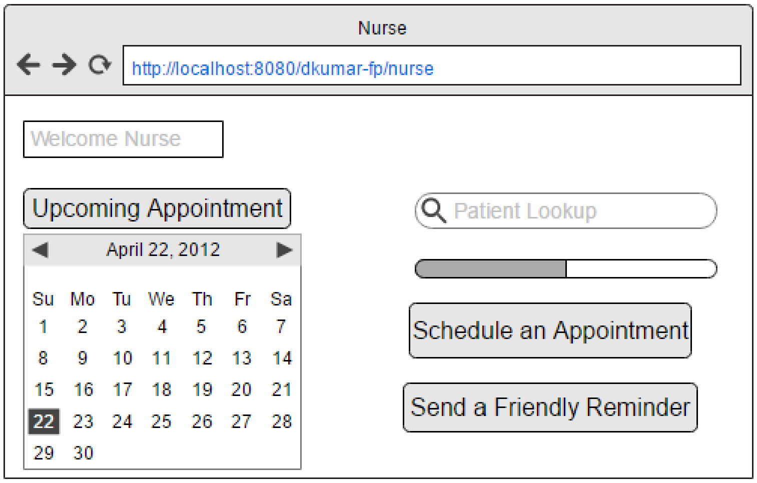




Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					



Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					



April 22, 2012

Su	Mo	Tu	We	Th	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30					