**Project requirements**

**This is an engineering proof of concept. It is not a planned marketable project. The goal is to exercise the Spring MVC technology according to “company” guidelines to validate its use in future projects.**

**The presentation should emphasize demonstrating the technical capabilities of Spring MVC. It should NOT be primarily a demonstration of the user interface.**

**The man hours spent on the project is expected to be ~100 hours for a 4 person team. This should be tracked in the daily README file and in a spreadsheet.**

* **Use the technologies we learned** 
  + Annotation
  + Data binding
  + Validation; Custom Validation
  + Custom Formatter
  + Uploading files
  + Internationalization
  + Exception Handling
  + REST/Ajax two way; error handling
  + Security ; Login, Authorization
  + Web Sockets
  + Web Flow
  + MVC Testing
  + Tiles
  + Persistence in database [JPA]
  + **Documentation should include EXACTLY where technologies are used.**
  + **Documentation should include EXACTLY how to test the functionality.**
  + **Documentation should include WHO did WHAT**
  + **DO NOT use technologies that we haven’t covered…**
* Other “technical” requirements include
  + Follow “corporate” architectural decisions
    - This isn’t a one-off design/implementation
    - Pivotal Server [Tomcat] based
    - MySQL based
    - Adhere to project structure as used in class
    - Adhere to configuration as learned in class
    - NO CSS libraries [ e.g. NO Bootstrap]
    - NO javascript libraries [ exception jQuery]
    - GitHub Manual usage…
  + Good naming practices [[Oracle naming](http://www.oracle.com/technetwork/java/codeconventions-135099.html) ]
    - Proper package naming
    - Descriptive names [ Product product .vs. Product p .vs. Product prod ]
  + Javadoc ready [\*internal note see Final1214]
* Every Team member is to “own” a “front to back” scenario
  + View, Controller, Domain Object, Validation, Service and Repository
  + **Documentation should include WHO did WHAT**
* Use design pattern of MVC and/or separation of concerns
  + Put the data manipulation in service layer
  + Put your data computation in service, and domain model[ invoked through service]
  + Do the data access in service through repository
* Good and organized pages
  + Header, footer, standard navigation, etc.
  + page template/consistency
  + Static pages about the company and contacts
* Design documentation
  + Class and sequence diagram, etc.
  + Assumptions, risks, dependencies
* Documentation should include how to configure /Install Application
  + database name should be **team name**
  + Application should have pre-populated data [ populate.sql].

**REQUEST:**

**BUT I want to use the Bootstrap CSS CDN… ANSWER: Follow the Guidelines**

**BUT I want to use the Angular JS library… ANSWER: Follow the Guidelines**

**BUT I want to use SQL Lite … ANSWER: Follow the Guidelines**

**BUT I want to use Jetty … ANSWER: Follow the Guidelines**

**BUT I want to use an automated email process … ANSWER: Follow the Guidelines**

**BUT I want to use Hibernate without JPA or Spring Data … ANSWER: Follow the Guidelines**

**BUT I want to work with my Teammates on EVERYTHING … ANSWER: Follow the Guidelines**

**BUT I want to use the Spring Boot… ANSWER: Follow the Guidelines**