Info for Capstone project:

Mission Statement should be 1 or 2 sentences.

Detailed Description should include any "obscure" info that can give us a visual of what it does, doesn't, can, can't do.

While you are free to use any technologies you wish, keep in mind that the further you stray from the content of the bootcamp, the more "on your own" you will be as far as peer support goes.

Since you will be giving a demo of your project, which of these phrases would you prefer to have your audience say about your project:

1). "That looked like a really cool idea, but since I was told what it was supposed to do, instead of actually seeing it, I don't think I understand the full experience. I would have loved to see it work correctly".

2). "After that 15 minute demo, and having the user docs available, I think I can use/navigate that app on my own to play with it a bit more".

Tentative presentation schedule (5 minute screen sharing session)(process walkthrough)(rough demo if you wish):

Wednesday 2/5/20 (Sprint 0):

Environment setup

User stories (requirements)

Product backlog (full list of requirements)

Identify "candidate" types (nouns) and actions (verbs)

Begin data dictionary (project glossary)

Initial (rough) documents: Flowcharts, ERD, UML, Use Case, Testing docs (TCERs), Wireframes, etc.

Friday 2/7/20 (Sprint 1):

Refine candidate types into required classes w/ properties, methods, relationships.

Structural organization (packages)

Skeleton Code

API documents

Capstone Project Requirements:

- Data maintained by some sort of persistence technology (MySQL, SQLite, MongoDB, etc.)

- Manage no fewer than 4 and no more than 7 database entities and their relationships. (Bridge tables are not entities.)

- Data access w/ JdbcTemplate or Jpa (Jpa may qualify as a stretch goal. More on that below.)

- Spring Dependency Injection

- Spring Boot

- UI is HTML, CSS, and JavaScript.

- UI may be rendered either client-side (@RestController w/ JavaScript) or server-side (Thymeleaf). At least one view must use client-side JavaScript rendering.

- Input validation, domain validation, **the app must never crash**.

- At least one stretch goal! -- Research and successfully implement a complex technology we haven't covered in class. Past examples include: Robust map integration, application caching, D3 (or any rich client-side graphic library), React/Angular/Vue, non-relational data stores...

- Authentication/Authorization

- Robust testing for all backend components.

- ERD for the database

- Workflow representation. Does not have to be a flowchart.

- Minimal Javadoc for class and methods.

Guidelines represent a minimum starting point. If you have a compelling case for alternative technologies: non-relational data stores, mobile development, hardware hacking, etc., then present it. We reserve the right to veto any and all variations from the guidelines. Therefore, projects must be approved before start.

Capstone project template (submit this page for approval)

Name:

Mission statement (what it is):

Detailed Description (what it does):

Stretch goal(s):