Final Capstone – Things to Think About or Consider

(04/2023)

1. Read all README files in their entirety and understand what they are telling you. Read them several times if you need to. Refer to them as you develop.

2. Ask questions of the Scrum Master and/or Product Owner if you don't understand anything. DO NOT ASSUME.

3. **Analyze. Design. Code. Reflect.**

4. Use the **App --> Service --> Controller --> DAO** application design pattern. Refer to the **Final Capstone General Application Structure.pdf** document if you need help in understanding how to do this.

5. Use the "Separation of Responsibilities" principle in your design. Do not allow a feature of your project to perform more work that it requires. You may end of with several smaller units of code where your initial thought it was it one.

6. Work on the "Happy Path" before edge cases, in fact, don't worry about any unnecessary or extreme edge cases at all. Edge cases such as SQL Exceptions, User input validation, Null pointer exceptions are about all you should be handling. Even then, handle them **AFTER** the "Happy Path" processing is complete (or if/when they "pop-up").

7. Remember, you can perform data validation on the Server-side as well as the Client-side. Some data validation might be better done by one side or the other.

8. Be sure you start the Java, back-end, API server before you run your Vue project. Verify that the server port is **9000**. If it is not **9000** the starter authorization code will not work properly.

9. **You should not have to make any changes or additions to the back-end starter code provided. Doing so could render them unusable.** Any code you write for the back-end should be in new files and or packages. ***If you feel you must positively, absolutely modify any of the back-end starter code, check with your instructor beforehand.***

10. Create new classes in the appropriate packages for your controllers, model and DAO components. Consider creating a new package for your JDBC DAO code.

11. Consider using pgAdmin to examine your tables in the **final_capstone** database.

12. Place any tables you may need in the **final_capstone** database,

13. **DO NOT** modify the **users** table created by the *setup* script.

    The **users** table is used by the security and authorization code provided. **If you change this table in any way you may render this code unusable.**

    If you need to keep information about a user of your application, create a separate table for that information. If information in the **users** table is required for an application user, consider creating a relator table between the **users** table and any the other table and/or use an SQL join/sub-query to get the related information from the **users** table you need.

14. Remember each member of the team needs to create the database and tables used in your project. Be sure to save the SQL needed to create and/or insert data into your project tables are stored in the **backend-server/database** folder of your project.

15. Consider using a free wire framing tool to lay out the general format of your Web User Interface (UI). Your instructor can recommend some, if you ask.

16. A major decision in your frontend is how to share data between components. Remember **props** should be used is a component has a need for a unique data value, and the **Vuex Data Store** for everything else.

17. Consider designating one team member the liaison between the team and Product Owner for communications such as questions and scheduling meetings. Being bombarded by several people from a team about the same subject does not make for a happy Product Owner.

18. **Use resources you already have available** when struggling with how to do something: **lecture-final** code, **exercise-final** code**, tutorial-final** code, **Documents/Artifacts** in the Student Resources Folder, session **recordings**. They contain examples of virtually anything you have to do and are available 24/7/365. We know it is easier to ask someone for help, but you will learn much more if you at least attempt to come up with the answer yourself. That's said, ask for help when you are unable to use the available resources to find a resolution to your issue.

19. Be careful that only one team member is editing a specific file or making changes to that file at a time. It's OK for one person to be working on one file and another person on another but be sure each of them has pushed their changes and pulled the others to stay in sync. **Two people editing the same file is sure fire merge conflict.**

20. **If you get merge conflicts or git errors, check with your instructor** if you are not sure how to fix them. Better to ask and do it right than think you know what are doing, do it and mangle your repository.
21. Come up with a way to coordinate who is working on what files to avoid problems. Comments on the Trello card for the part of the project you are implementing is a good place for this.
22. **Commit and push often.** Consider coordinating team pushes (and pulls) throughout the day. Communicate to your team when you have committed and pushed any changes. Pull any changes made by your teammates as soon as possible after notification.
23. Be sure you have pushed any and all changes you have made before you go to bed or will be unavailable and let your team members know what you have pushed. Meaningful commit messages go a long way to help your team understand what you have been working on and where you are.

24. **Do not make unilateral changes to any part of your project**. Involve your teammates before you start any changes.

25. Remember, **you are a team of peers**. No one person is in charge and all team members should contribute to the project solution. Everyone's opinion, comments and ideas are valuable. If you feel you are being marginalized by your team or otherwise not having a good experience, let your instructor know.