Franks Comments Regarding the Module 2 Capstone

If you get stuck, look at code you have coded that works or code you were given for guidance. Everything you need to do in the capstone you have seen and used, many time, in exercises for examples. The **14_Server_Side_APIs-Part-2** ***lecture-final*** has good examples of what you need to do on both the client and server sides.

Refer to the **Module-2 Capstone Design** document in your Student Resources Google Drive (and project folder) folder to help you understand the existing project design.

Remember:

- **The required use case code for the client side has already been completed. <u>There is nothing for you to do in it.</u>**
- You will be working on the server side only implementing the data access and server side APIs.
- Use Postman to test your server side code.
- You may try running the client to test your server side code ***AFTER*** you have verified it is correct using Postman.
- Review all the code you are given in the starter project.
- Don't forget to create the **tenmo** data base & run the SQL to create tables load them on each team members computer.
- **Server** - Coordinates requests and data source interactions. It receives a request, finds the appropriate data using a DAO method and returns. The client services methods will use the paths defined for the controller methods to make their API calls,
- **DAO** - Manages data source access. Interacts with the serve to provide data to the client side when requested. Nothing unrelated to data access should be in the DAO. Test your SQL in pgAdmin before using it in the program.
- **Think in objects.** objects are passed between methods and the method decides what to extract or change using the object's getter and setters.
- **Logic and processing on the server side** should be done in a DAO whenever possible, NOT the controller. For example, the server side should NOT be calculating the Account balances for a Transfer. That should be done on the client side. All the server side should do accept the Accounts objects with the balances already adjusted and send them to the DAO method that will update the database using the Account objects it receives.
- **If you have MapRowToObject methods** in your DAOs, be sure all columns required by the method have been included in the *SqlResultSet* object you send to the method. If not, you will receive an **Invalid column** error message on the server side when the method is used.
- **Run the client code and register a couple of user** to get some accounts in your database for testing.
- **Use the "TODO"s** in IntelliJ project files to help determine what you need to do.
- **Ask questions** if you are unsure about what to do. It is better to ask and be sure, than assume/guess and be wrong.