Coding Guidelines for Java/Spring submissions

Assignment and Instructions:
A retailer offers a rewards program to its customers, awarding points based on each recorded purchase.

A customer receives 2 points for every dollar spent over $100 in each transaction, plus 1 point for every dollar spent over $50 in each transaction
(e.g. a $120 purchase = 2x$20 + 1x$50 = 90 points).

Given a record of every transaction during a three month period, calculate the reward points earned for each customer per month and total.

- ***Make up a data set to best demonstrate your solution***
- ***Check solution into GitHub and share the URL(Make it Public)***
- ***Solution should be built using ReactJS or Spring(Spring Boot)***
- ***Instructions for running the solution should be uploaded in ReadMe file on GitHub***

Mandatory

- Must expose RESTful endpoint – for accepting a customer id and returning reward points [At a minimum] . Refer to the assignment for calculation logic of reward points

- Do not use hard coded/magic numbers in the code.
- Ensure there is no unused variables in the project; that demonstrates the coding style. Try to close warnings from the project.
- Calculate awards points from the purchase amount. Don't exclude cents/decimal part while calculating.
- API responses should return appropriate payload and HTTP code for both Success and failed requests
- Create appropriate test data covering scenarios
- Include appropriate test cases
- Avoid pre-requisites like DB etc. Use In memory DB or other in memory option for data
- Readme file is mandatory – must contain the steps required to build and run/test the code
- The solution must be checked into Github (provide a public github url)
- Binary Files should not be checked into github
- Calculate awards points from the purchase amount. Don't exclude cents/decimal part while calculating.

The following are nice to have and demonstrates our knowledge of the framework

- We can also add an Exception Handler which serves as a catch all and send back INTERNAL_SERVER_ERROR in case of uncaught exceptions
- In Spring, since the framework creates a proxy interface, we do not have to create an interface explicitly especially if there is only one implementation.
- Try to use lombok for pojos, since the definition of a pojo comes across more precisely and it also provides useful annotations for the log4j bridge for logging
- Functional Programming constructs of Java 8 as appropriate.
- Please ensure to mention in Readme about build dependencies. Explicitly mention if its Maven or Gradle project.