1. **Learn and be Curious**

* I like learning in my free time and keeping myself updated with new technologies and frameworks that come up in the software industry. The sources I use are mostly websites and blogs like baeldung, geeksforgeeks, etc. I also enjoy following youtube video series that explore a particular technology/framework over the course of 10-15 episodes. It helps me kind of play around with and delve into the tool, and I then try to follow the advanced articles and videos on the topic when I find time later. I also like to read a lot of Stack overflow posts when I research a solution to a particular kind of problem. I also like to enroll in courses/certifications on platforms like Coursera/Udemy from time to time.

1. **Insist on Highest Standards**

* **Problem**: For any feature implementation, I follow a process for ensuring the code I write is properly tested and automated before we deploy to production. I was once working on a high priority change which involved integrating external client APIs on a project, which needed to go to production in a short amount of time. I was told by the design team and senior developers to finish the code development quickly, and as long as the minimum functional changes were validated by testing team, that it would be good from our side.
* **Action**: I decided that I would finish the code changes as quickly as possible, so that I could squeeze in some additional time to focus on the testing part. I made sure to ensure the final code changes which went to production were 100% covered in test scenarios in my code, and I worked with the testing and automated teams to ensure that the integration changes were completely validated and automation test suites were created. It took some extra calls and some short lunch breaks, but I got it done.
* **Result**: When the code changes finally went to production, I did not encounter any issues, and I was glad that the due diligence I had put in earlier with respect to complete testing ensured there was no rollback in changes and the integration happened seamlessly.

1. **Invent and Simplify**

* **Problem**: In the app that I work on at my present company, earlier we only used to have a couple of APIs to return the terms and conditions for data sharing to be displayed to the user on the app. But as new and specific products/features were released, the T&C for each such product and service had to be more specific to the nature of the product. Also, we needed to provide user the option to change their choice at a later point in the app, should they chose to do so. The initial requirement from the business team was only to minimally enhance the existing APIs to include for the T&C for new products.
* **Action**: I did an impact analysis of the future incoming requirements and along with the senior developer that I was working on, came to the conclusion that if we did not restructure the backend microservice involved to make it flexible to new changes, we would have a very tough time to implement new requirements that would come up later. We decided this needed a complete redesign of the backend APIs to support the consent flows in the app. I worked with the UI and UX teams to clarify future goals to restructure the backend microservice to return T&C for a customer/vehicle based on a variety of parameters. Also, any updates in this content received from the legal team (verbiage and images shown) could be updated with minimal effort at any point of time.
* **Result**: The redesign proved to be very useful and we were later able to incorporate a number of new consents based on vehicle capability or products. The data model I built for storing the information for a consent also proved to be useful for the overall app design and was later used in other microservices.

1. **Customer Obsession**

* **Problem**: In the app that I work on, I think of any feature I work on as being experienced by a real customer at some point in time. I often visualize if I was that customer, what would be my expectations be of using it?
  + how would I react to it?
  + Is the business transaction flow simple?
  + How simple are the instructions and screens to follow?
  + How responsive is the UI?
  + How would I feel if the app throws an error when I click a button or tap on a screen, or I’m unable to complete a transaction?
* **Action**: Before I start working on a feature change, I make it a point to work closely with the design and business teams on the
  + scope and impact of the feature
  + details on how the feature impacts user experience
  + concerns and issues that we might encounter in the future
  + how to make the feature as customer friendly as possible
  + having a clear testing and automation plan before the feature goes live
* **Result**: I notice that when I diligently follow the above process, I mostly do not encounter issues when a feature goes live to customers. It is a satisying and motivating feeling that I like to incorporate into my software engineering process.

1. **Ownership**

* **Problem**:
* **Action**:
* **Result**:

Learn and be curios – generic, hybrid pulse

Insist on highest standards – testing, clean assist

Invet and simplify – dataconsent service

Customer obsession: data heavy APIs, demo content

Ownership - Hybrid pulse testing

Have backbone, disagree and commit – owners manual

Deliver results –

Earn trust – ota notification, testing