# Best practices

1. Every story should have BDD

Whoever completes a story they should write BDD, else know body knows who did what

# Design wise

1. Always code using flag based/Toggle functionality - that functionality should be disabled easily

Ex:- use @ conditionalOnFlag to enable and disable

* we have flag for enabled-disabled batch code using true/false,
* we have flag for elastic APM,
* Flag to enable and disable pops
* Ecar also have many flag based toggle functionality

1. Try to externalize all the properties, - if sriram missed to extenalise elastic apm url, then we should have made code change and redeployed everything

## Coding wise

1. We should have proper retry mechanism
2. Mandatorily use kafka –producer retry mechanism
3. **Implement kafka transactions-** like both Fiserv call and sending kafka messages must go hand in hand, either all should be success or either everything fails
4. All rest service interactions should be with another layer and in another project like FIT, 1DSTR in both the proj’s all
5. All exception handling in another project
6. Rest API methods should return only Response entity and all exceptions must be handled in that rest project only, they should not get cascaded to other projects – consider 207 multi status
7. **Implement caching mechanism and avoid hitting database / rest service multiple times**

# Standards

1. rest service request and response should be logged – use some rest interceptor
2. use mongo appender and log request and response into db whenever any exception came
3. always monitor overall performance- every story should have over all request processing time
4. implement logback and masking sensitive information

## Things to learn

1. Spring oauth
2. BDD
3. Junits revise
4. splunk

## New project onboarding in wells Fargo

1. Jenkins- Sonar Qube
2. PCF
3. Apigee why?
4. App dynamics on boarding
5. Venafi certificates-raise ticket
6. U deploy
7. Splunk on boarding
8. Kafka on boarding
9. Create mongo db access id
10. How are we getting data or do we have any screen
11. Is Venafi enough or is anybody else is going to use our application if any one else is going to use our application, how are we going to implement the security, do we need to use oauth –auth server, resource server .. something like that
12. Don’t log sensitive data – implement regex patters and all
13. Implement hard gating in git hub for minimum 80% code coverage with proper asserts
14. Spring cloud gateway
15. Spring security – jwks ,jwt – authorization server, oauth2.
16. P2dep
17. Configure mtls and non-mtls url
18. Risk team approvals – account number mask

Json max obj size -

Venafi certificates

We should hit with certificate

Set venafi credentials in headers

Apigee

Siva

P2DEP