# Bill Pay Application

# Object

XYZ Corporation would like to create a new bill payment application to act as hub between customers and billers. XYZ Corporation will maintain customer account with balances in it, will maintain biller list along with bill data and will store all transaction data.

# Common requirement

Provide a feature to “register” customer for paying bills. As part of registration capture email id only. Successful registration should create a “stored value account’ (wallet) associated with the registered customer.  Option for customer to move funds to the wallet as part of registration.
Pay your utility bills:
1.Fetch billers and bill from 3rd party external applications.
2.Select a biller, fetch the bill and pay that using the wallet.
3. Move funds from customer wallet to biller account.

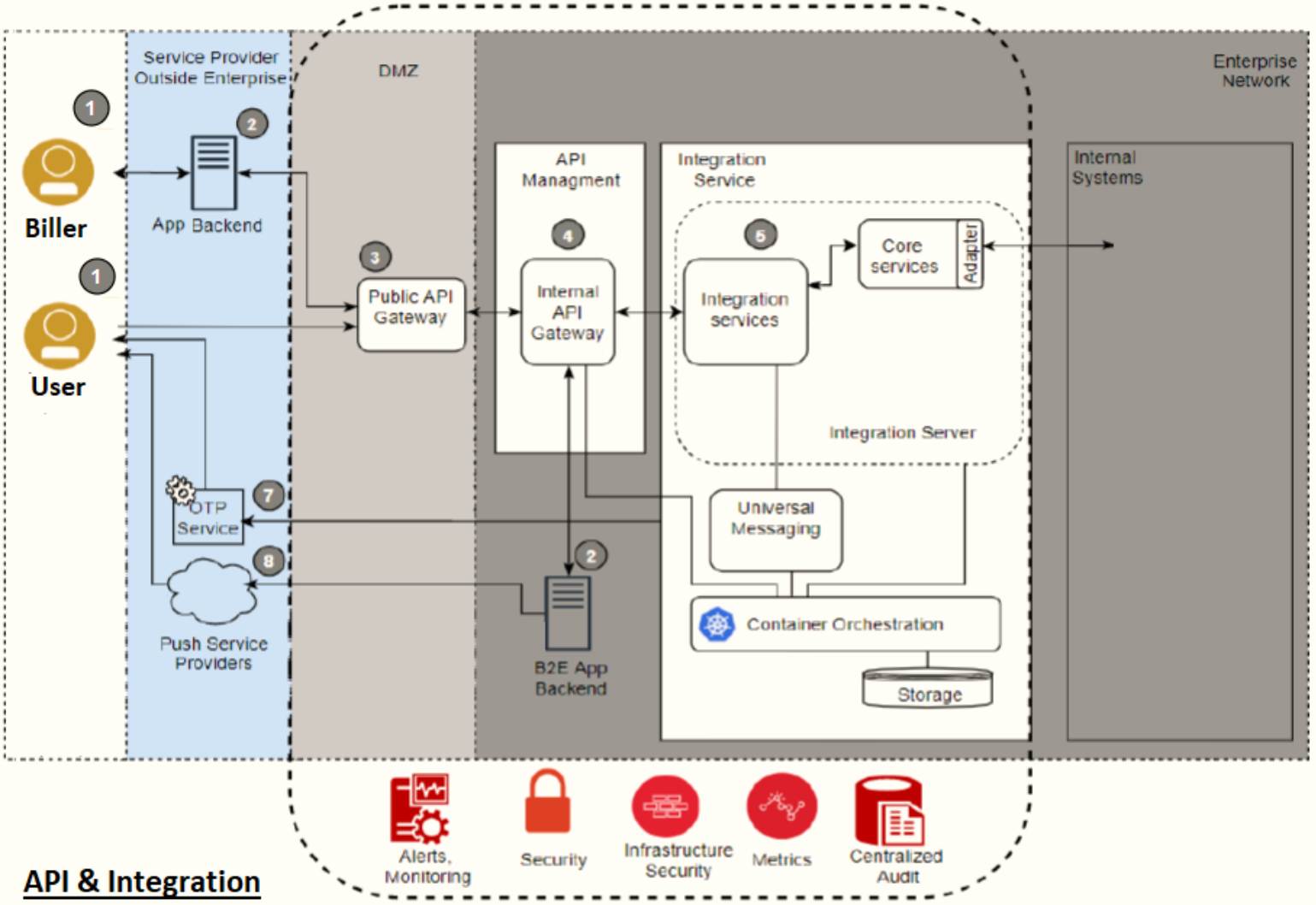
###Bulk/File based Bill Payment Process bill payments for N number of records received in a file. Multiple files can be received in a day. Design should support following use cases

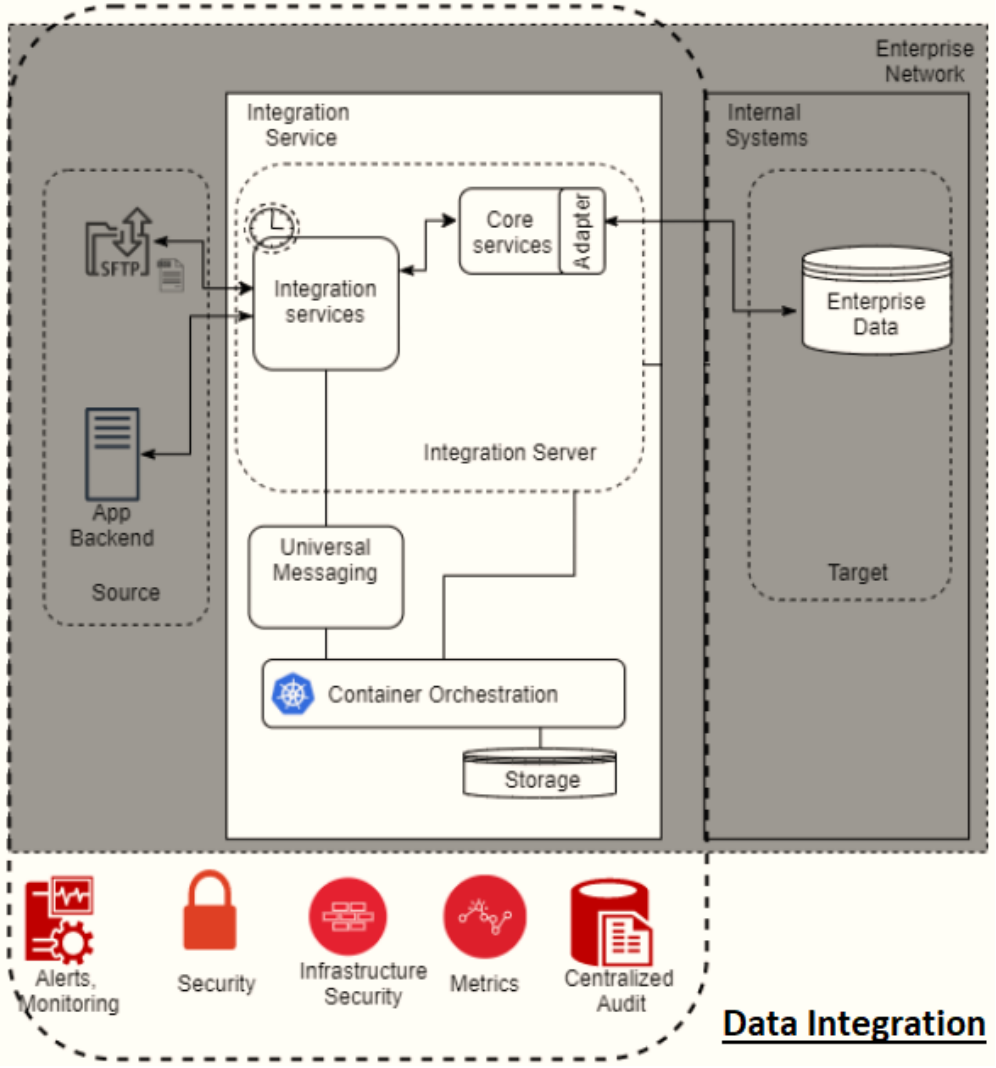
# Technical requirement

Requirement 1 - Design assuming XYZ provides a standard specification for Inbound bill payment file and customers (corporates, banks, service providers) adheres to this format when sending the bill pay file.

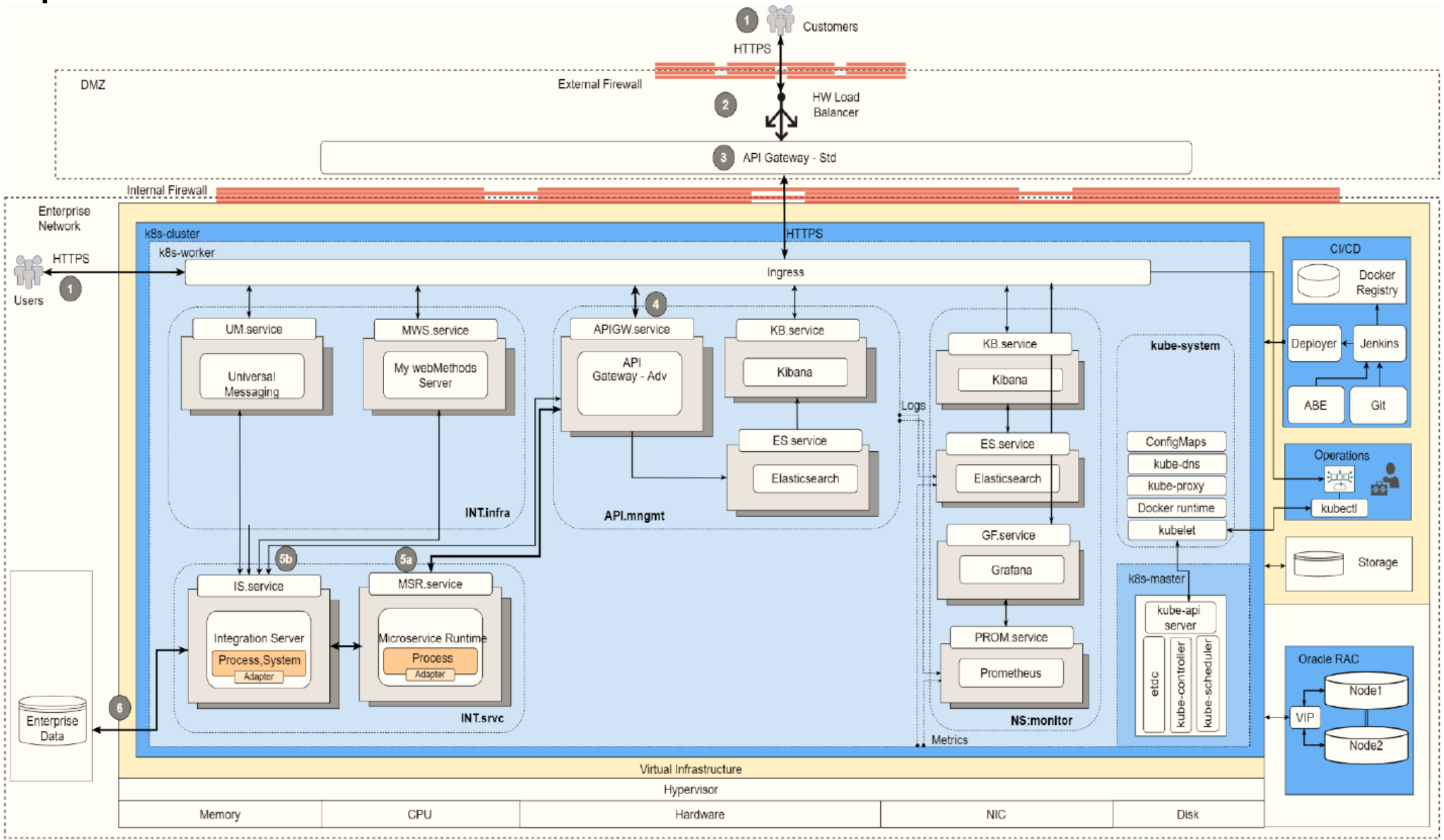
Requirement 2 - Design should also support non-standard format / ad hoc format in future. The design should require minimal code changes to support new file formats introduced in future.

# API & Integration and Data Integration

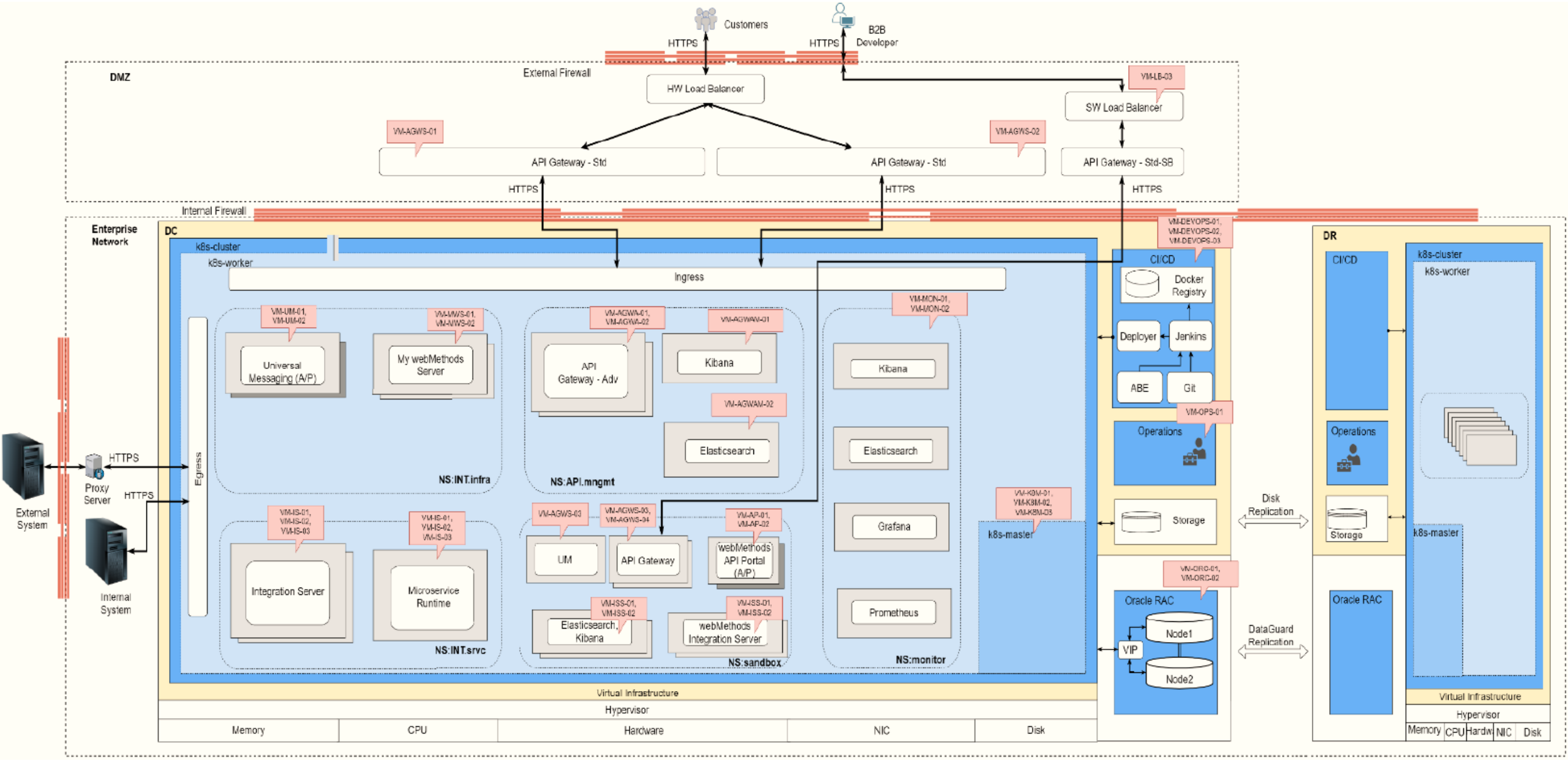


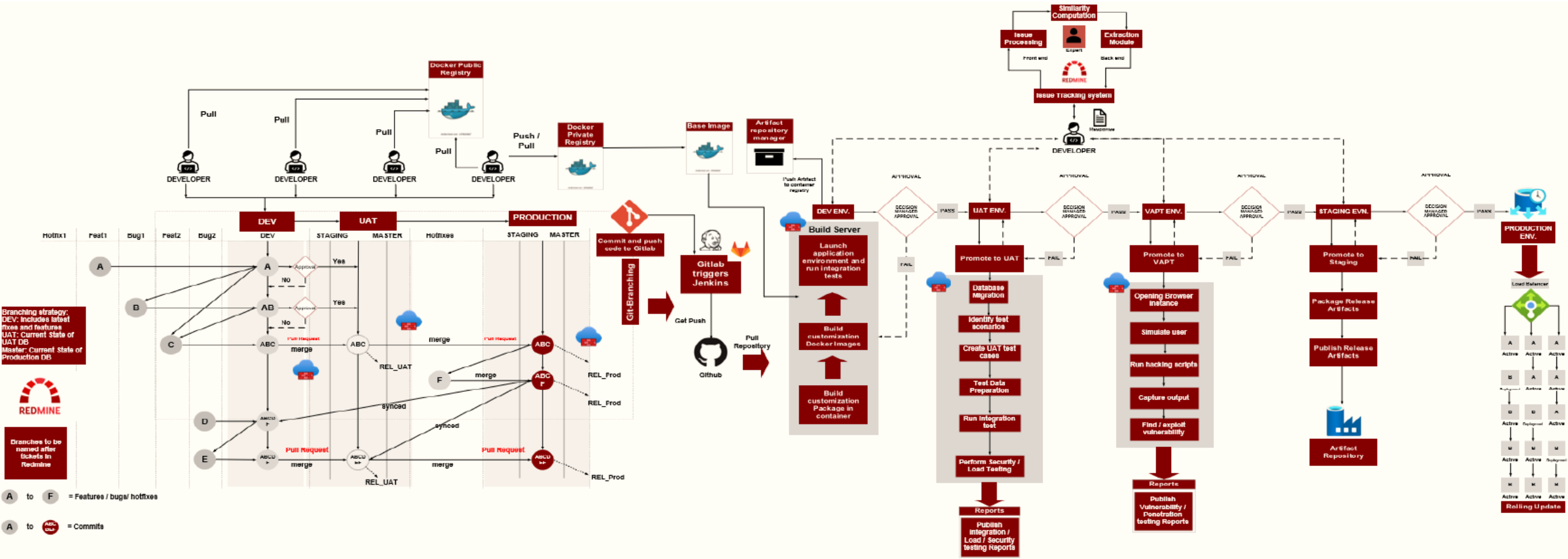
# Component Architecture



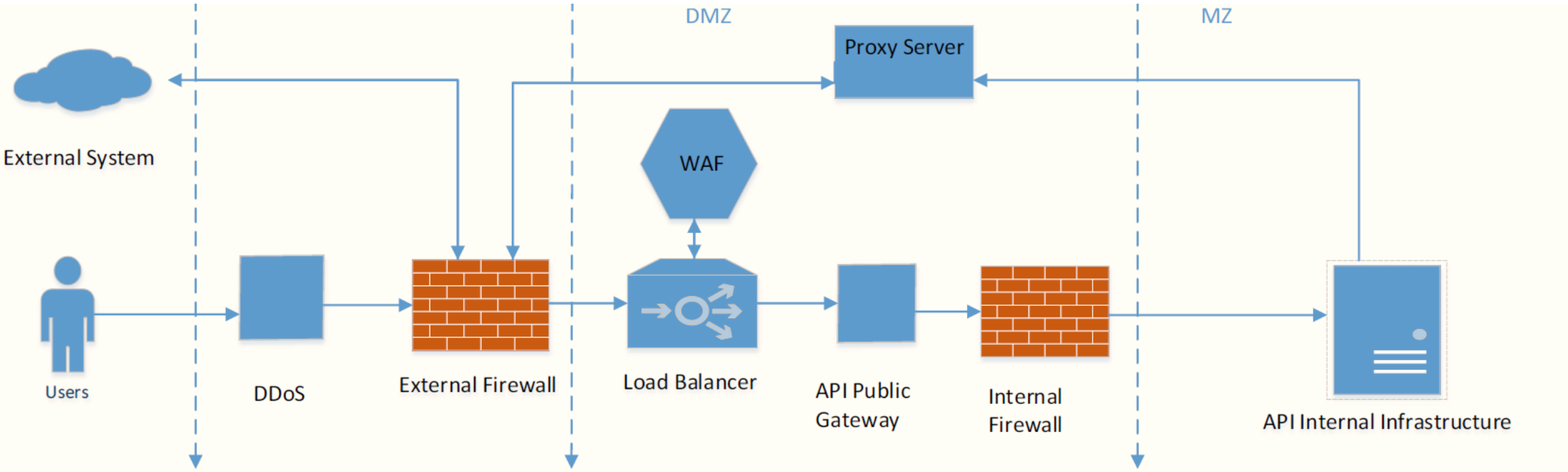
# Prod - Deployment Architecture



# DevOps Architecture



# Security Architecture




DDOS protection

SSL offload on load balancer

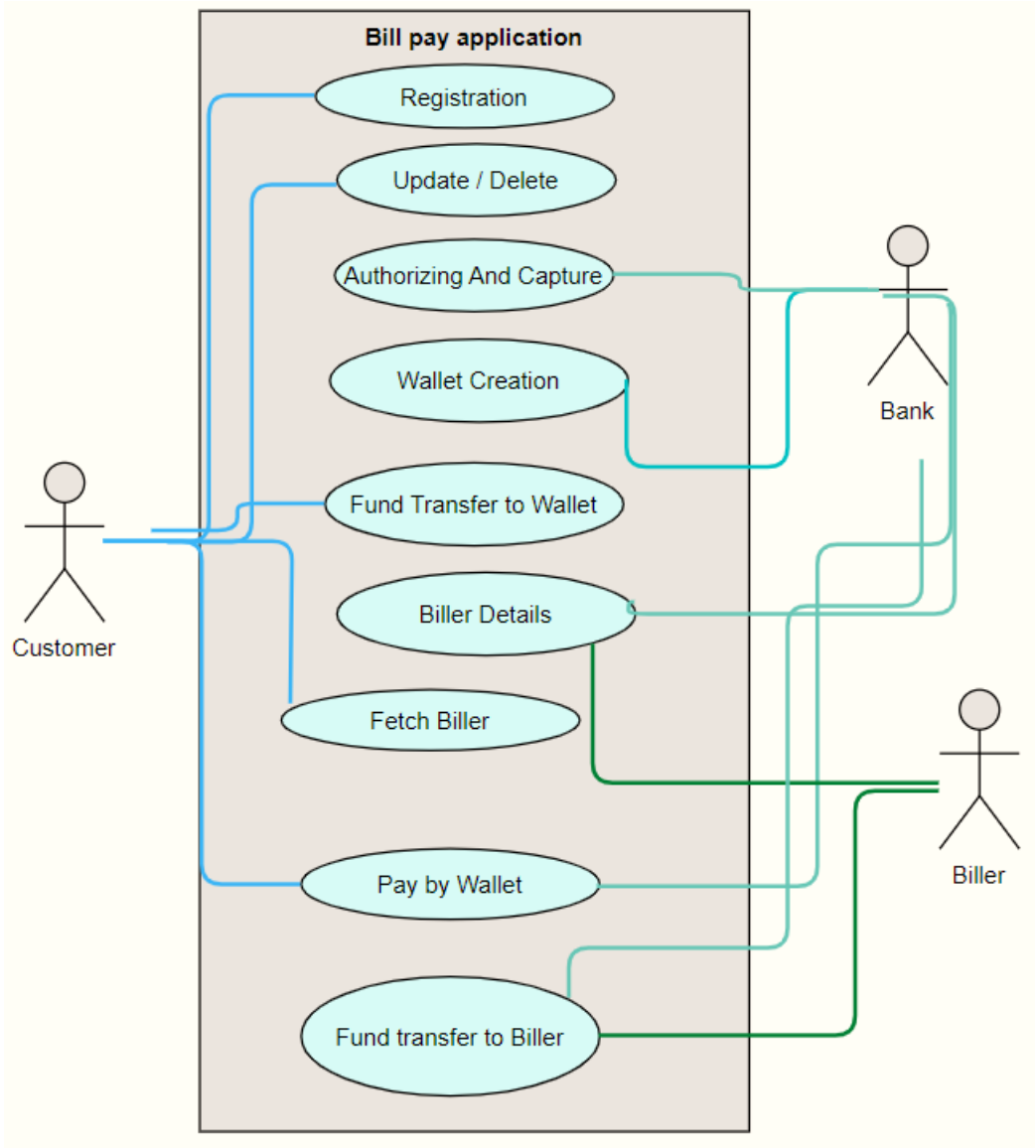
WAF integration to deal with application layer attacks

Proxy integration for outgoing URL based APIs

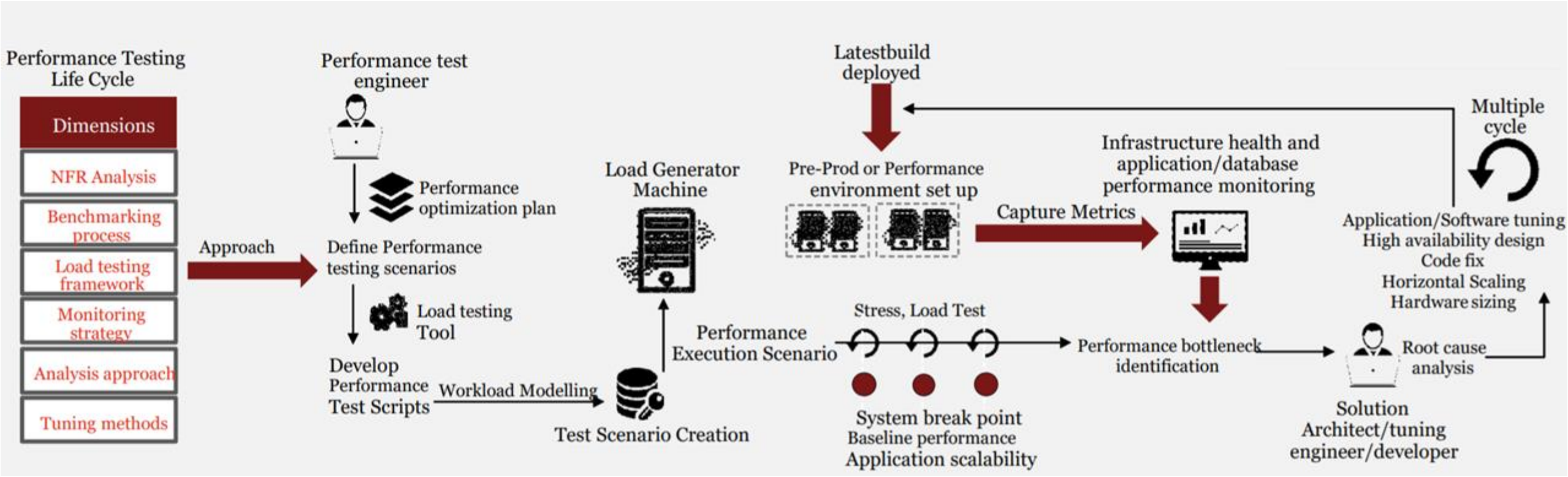
Natting at external firewall for IP based communication

API baseline

# Use case


Bill pay application
Registration
Update / Delete
Authorizing And Capture
Wallet Creation
Fund Transfer to Wallet
Biller Details
Fetch Biller
Pay by Wallet
Fund transfer to Biller
Customer
Bank
Biller

# Testing approach



# NFR

| Parameter | Description | Additional Details | Default Value |
|---|---|---|---|
| **Timeout** | This property is used to set all timeouts on an application-specific basis | The timeout will be different for different as per API category as follows:<br>• Retrieval APIs<br>• Financial Transaction APIs<br>• Non-Financial Transaction APIs | Financial APIs – 20 seconds<br>Non-financial APIs - 30 seconds<br>Retrieval APIs – up to 1 min |
| **Throttle/Rate Limiting** | The number of API calls an app or user can make within a given time period | Throttle/Rate limiting will depend on the API/User category as define by the business. | Basis throttle limit will be decided |
| **TPS (transaction per second)** | This gives the number of transactions per seconds that the API can handle | This parameter will be different for different APIs basis business's categorization for the API.<br>E.g.: A payment API will be called for wallet top-up more than a registration API hence TPS for payment API should be more than TPS for registration API | 1000 |
| **Scope** | This tells which customer roles can access the API | Business will need to identify customer roles for each API | As per business |
| **Max Request and Response size** | This gives the maximum size of a request and response in KB | | As per business i.e. > 50 KB |
| **Latency** | The total amount of time taken by the system to respond to the API call | The latency is only at the gateway layer in isolation.<br>Total latency will be latency of API gateway + latency of underline application. | As per business i.e. - 200-300ms |
| **Reliability percentage** | This refers to the probability that the system will meet certain performance standards in yielding correct output for a desired time duration | | As per business i.e. - 99.9999% |
| **Number of retries** | This parameter gives the maximum number of retries which will be available for the API | This value can change for specific APIs as per requirement. | As per business i.e. – 3 |
| **Availability** | This defines uptime for the API gateway | | As per business i.e. - 98.98% |
| **Messaging between the system** | Passing the data between the channels | This should be according to enterprise | As per business i.e. - ISO |
| **Communication Protocol** | Communication protocol amongst the system | This should be according to enterprise | As per business i.e. - RPC |
| - | - | - | - |
| - | - | - | - |
| - | - | - | - |

# API blueprint

| | |
|---|---|
| **API ID** | API xx |
| **Use Case ID** | Customer Service (Requirement Use case ID :xx) |
| **Module** | Accounts Register |
| **Overview** | Customer can register themself to the 'Bill payment' system by making an API request and receiving the success / failure response. |
| **API Description** | This API will be utilized by internal channels to register a customer account |
| **URL Endpoint** | {domainName}/api/{version}/registration/accountCreation |
| **API Type** | POST |
| **Associated Process** | This API will be triggered when the application wants to register customer's details. Customer's email id along with other details need to be sent as request to this API |
| **End User / Source System** | Mobile, Internet, BoT, Chat etc |
| **Downstream system and Target Service** | Business call |
| **Parent APIs** | Business call |
| **Child APIs and Sequence** | Business call |
| **Reference Document** | Business call |
| **-** | - |
| **-** | - |
| **-** | - |

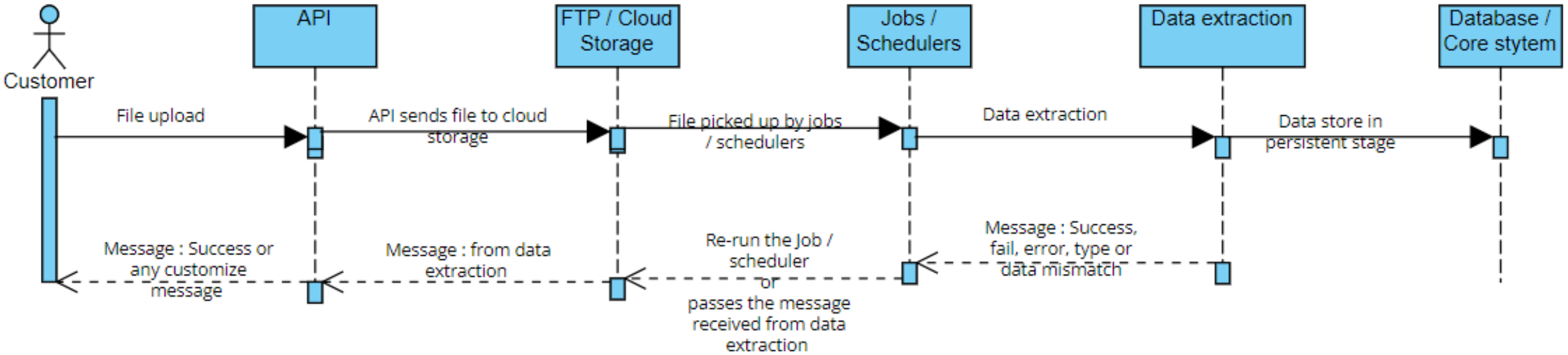
# API blueprint (request parameter)

| Field Name | Field Description | Field Length | Field Type | Mandatory | Sample Values |
|---|---|---|---|---|---|
| request parameter 1 | request parameter 1 | 12 | String | Mandatory | Name |
| request parameter 2 | request parameter 2 | 14 | String | Mandatory | ID number |
| request parameter 3 | request parameter 3 | 8 | String | Mandatory | email |
| channelId | Acquiring Institution Identification Code | 11 | String | Mandatory | 504511 |
| businessDate | Capture date | 28 | Date | Mandatory | 20/01/2023 |
| geolocation | Capture location | 50 | String | Mandatory | Latitude: 19.0188899 Longitude: 73.02894 |
| localDateTime | Local transaction data and time | 38 | DateTime | Mandatory | 2023-01-20 08:26:49.219717 |
| - | - | - | - | - | - |
| - | - | - | - | - | - |
| - | - | - | - | - | - |

# API blueprint (response parameter)

| Field Name | Field Description | Field Length | Field Type | Mandatory | Sample Values |
|---|---|---|---|---|---|
| Message | Request status | 50 | String | Mandatory | Success or failure or error |

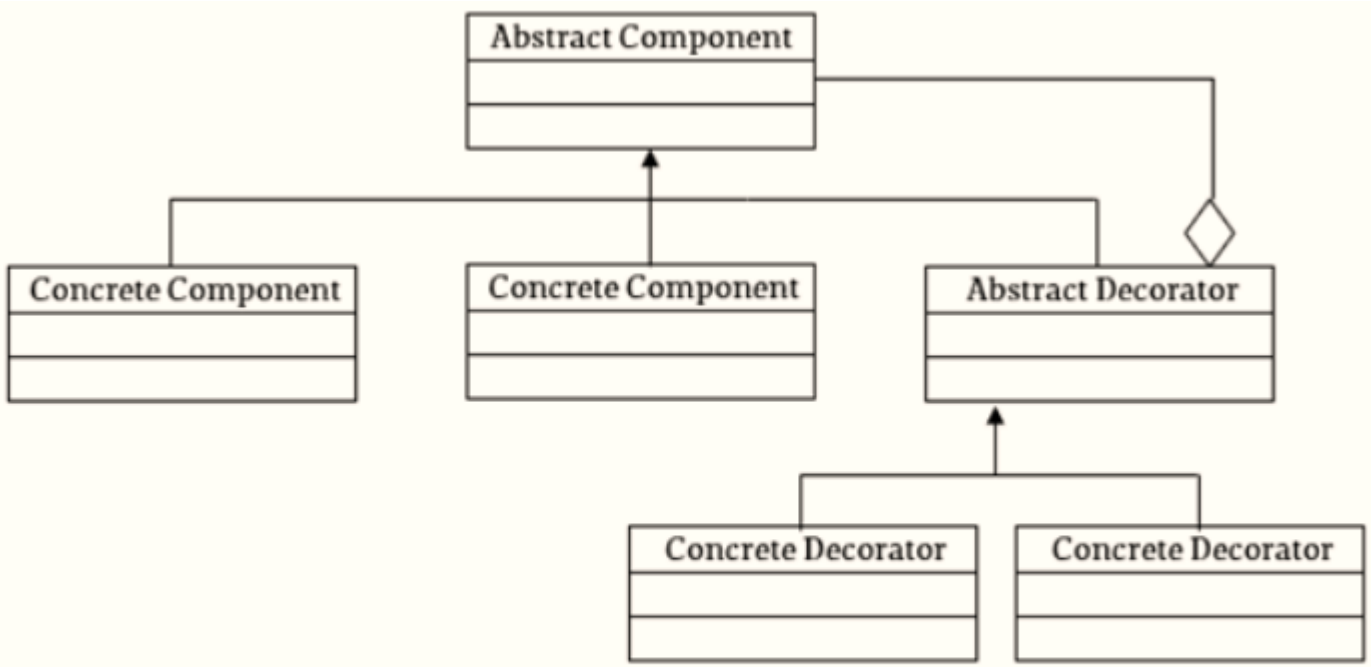
# Sequence diagram for file upload (requirement 1)



# Class diagram to support non-standard formats (requirement 2)

## Decorator Design Pattern

**Decorator Design Pattern** - With the help of 'Decorator Design Pattern' we support non-standard format / ad hoc format in future. This design will require minimal code changes to support new file formats introduced in future.

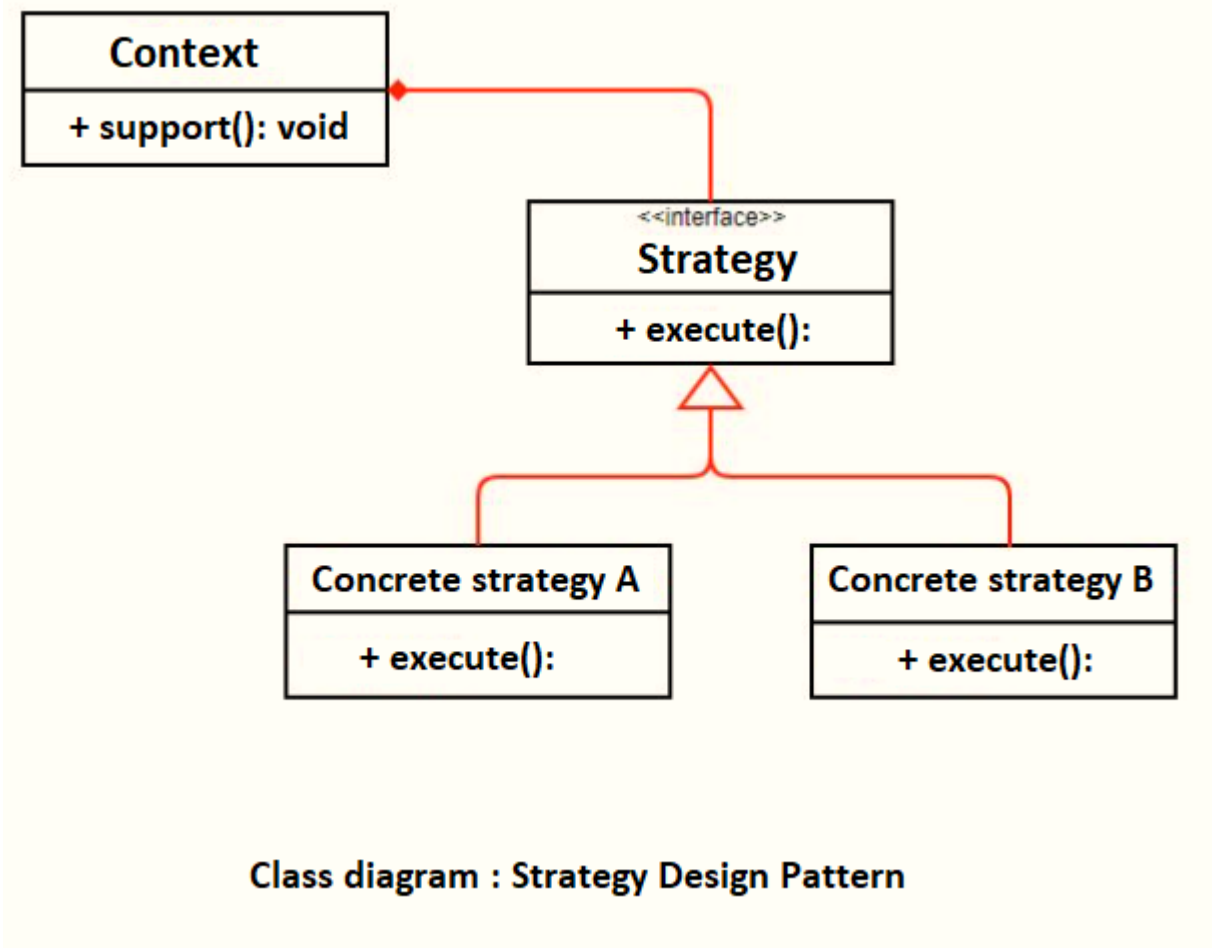



Class diagram : Decorator Design Pattern

# Class diagram to support non-standard formats (requirement 2)…

## Strategy Design Pattern

**Strategy Design Pattern** – This design pattern is also capable to support the non-standard format / ad hoc format in future. This design will also require minimal code changes to support new file formats introduced in future.

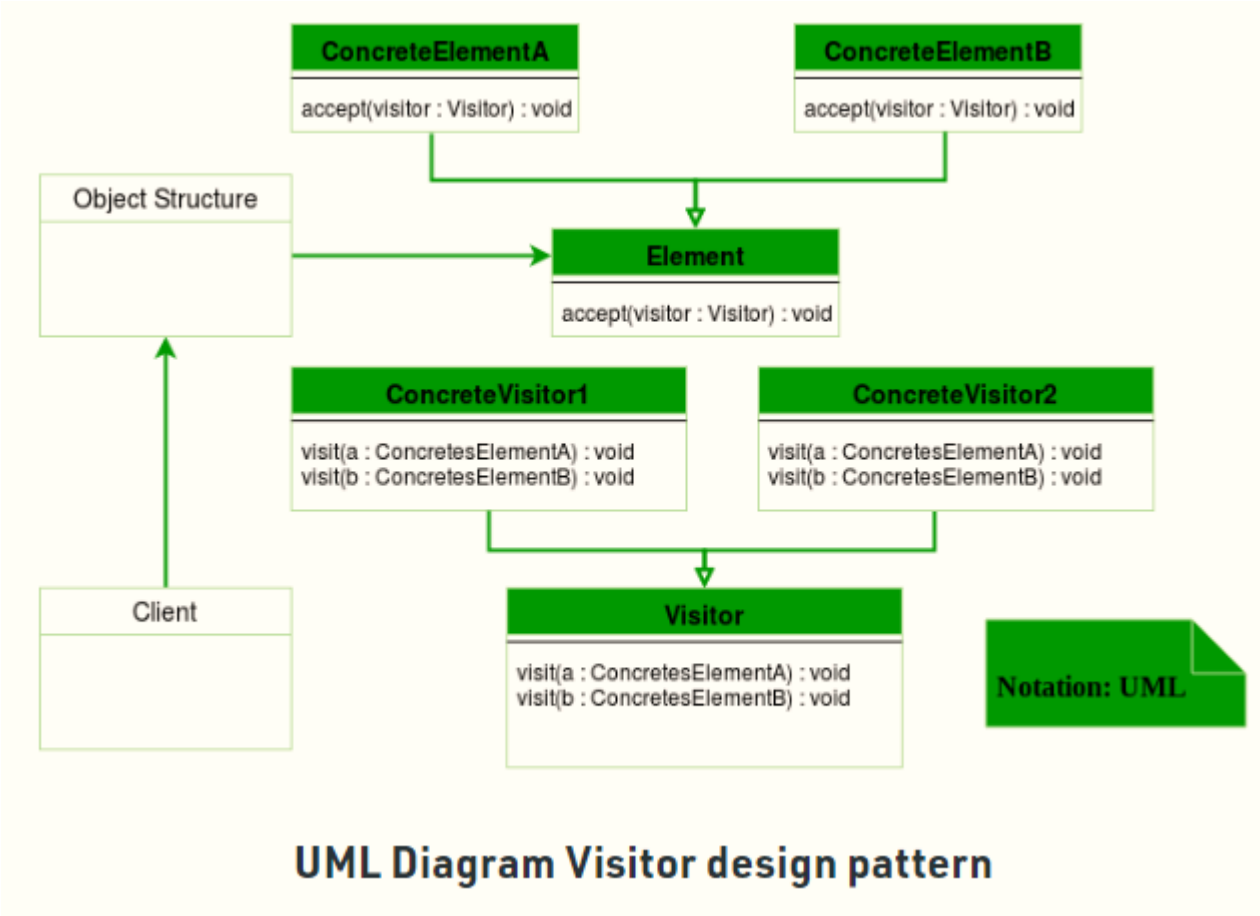



Class diagram : Strategy Design Pattern

# Class diagram to support non-standard formats (requirement 2)…

## Visitor Design Pattern

**Visitor Design Pattern** – This design pattern is think of to support the non-standard format / ad hoc format in future. This design pattern will also require minimal code changes to support new file formats introduced in future.




UML Diagram Visitor design pattern

# Notations used

| | |
|---|---|
| A | Alphabetic characters A through Z and a through z |
| N | Numeric digits 0 through 9 |
| P | Pad character i.e. space |
| S | Special Characters |
| An | Alphabetic and Special Characters |
| Ns | Numeric and Special Characters |
| Anp | Alphabetic, Numeric and Pad (Space) characters |
| Ans | Alphabetic, Numeric and Special Characters |
| MM | Month, 01 through 12 |
| DD | Day, 01 through 31 |
| YY | Year, 00 through 99 |
| YYYY | Year, 0000 through 9999 |
| Hh | Hour, 00 through 23 |
| Mm | Minute, 00 through 59 |
| Ss | Second, 00 through 59 |
| B | Binary representation of data |
| ..nn | Variable length data up to nn characters. There will be two or three character length (depending upon whether maximum data length is 99 or 999) at the beginning of the element to identify the number of positions following to the end of the data element |
| DC | Data Centre |
| SAF | Store and Forward File |
| PBF | Positive Balance File |
| SOL | Service Outlet |
| - | - |
| - | - |
| - | - |

# Functional and Error code

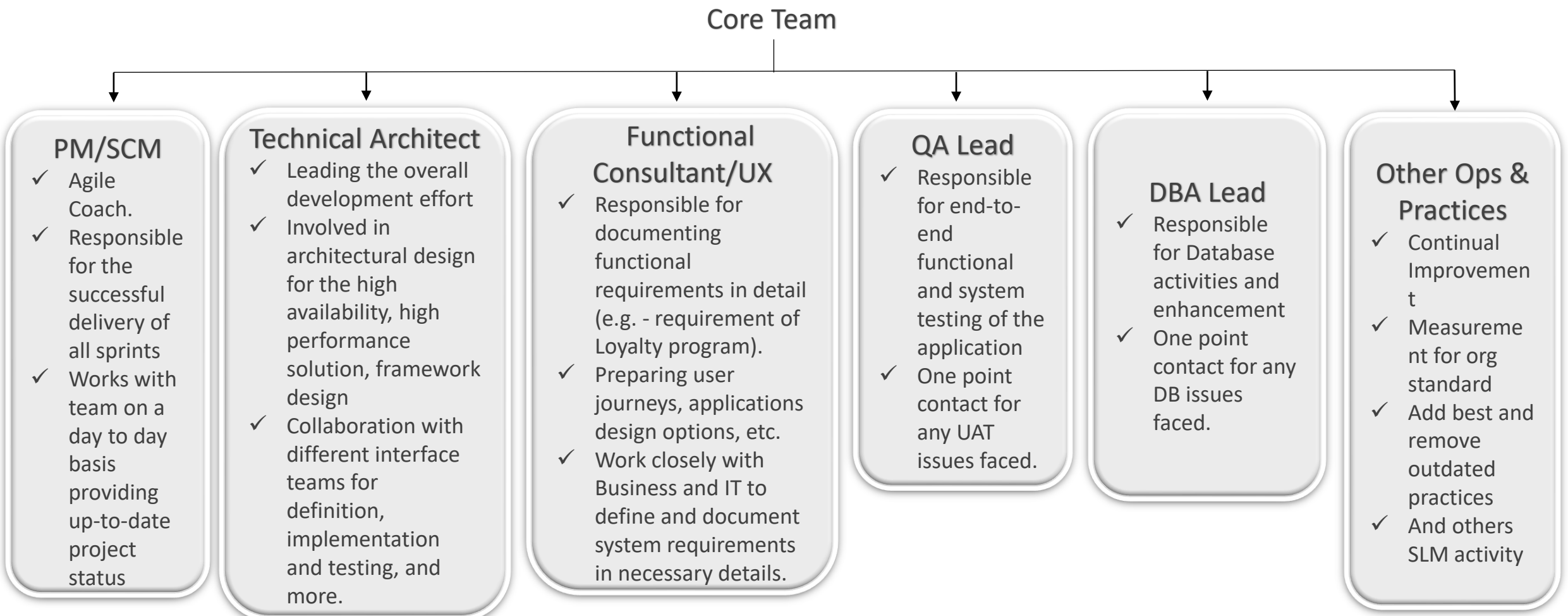
| Code | Meaning |
|---|---|
| 200 | Original Authorization request or advice |
| 400 | Reversal |
| 801 | Logon |
| 802 | Logoff |
| 831 | Echo-test |
| 881 | Others |
| - | - |
| - | - |
| - | - |

| HTTP Error Codes | Error Code | Error Description |
|---|---|---|
| 500 | E999 | Runtime Exception : <<Error Details>> |
| 200 | E000 | Success |
| 503 | E001 | End system <<System Name>> is Unavailable |
| 422 | E002 | Input data validation failed : <<Validation Error>> |
| 422 | E003 | Output data validation failed : <<Validation Error>> |
| 404 | E004 | No data found |
| 504 | E005 | Timeout error |
| - | - | - |
| - | - | - |
| - | - | - |

# Participating tool's list

| Project Management tool | Infrastructure Tool | Development Tool | Quality Management Tool | Release Management Tool | Risk Management Tool | Analytical tools |
|---|---|---|---|---|---|---|
| Jira / WorkZone / Trello / Redmine / MS Project etc | Cloud subscriptions i.e. AWS, AZURE, GCP, IBM etc | SDK | Unit testing tool i.e. J-Unit, N-Unit etc | Release Management Tool i.e. Chef, Octopus Deploy etc | Risk Register | Log Monitoring tool i.e. Splunk, ELK, Grafana, Fluentd etc |
| Document Management tool i.e. SharePoint, Jira, Google Drive etc | Code repository i.e. GitHub, BitBucket etc | JDK | Funtion testing tool i.e. Katalon, Selenium etc | API Management tool i.e. Apigee, Kong, MuleSoft | Root Cause Analysis | Audit Trail tool i.e. HighBond, TeamMate etc |
| Modelling tool i.e. Archi, MagicDraw etc | Cache Server | Database i.e. Oracle, SQL, Postgre and related tool to read and manipulate the data i.e. PL-Developer, Toad etc | Load testing tool i.e. JMeter, SOAPUI etc | | SWOT Analysis | |
| Communication tool i.e. MS Teams, Slack, Google Meet etc | DevOps Tool i.e. Jenkins, TeamCity, Circle CI etc | IDE i.e. MS Code, IntelliJ etc | Performance testing tool i.e. LoadRunner, LoadNinja etc | | Probability and Impact Matrix | |
| Resource management tool i.e. MS Project, Resource Guru etc | Container Orchestration tool i.e. Kubernetes, Helios, Centurion etc | Code coverage tool i.e. N-Cover, OpenCover etc | API testing tool i.e. Postman, Hoppscotch etc | | Brainstorming | |
| Collaboration tools i.e. Confluence, GitHub etc | Artifact Repository i.e. Nexus, CloudRepo, Yarn etc | Unit testing tool i.e. J-Unit, N-Unit etc | Code quality tool i.e SonarQube, Crucible etc | | | |
| | Load balancer | | Bug tracking tool i.e. Bugzilla, Redmine etc | | | |
| | Virtual Machine i.e. Linux, Windows etc | | | | | |

# Delivery Model



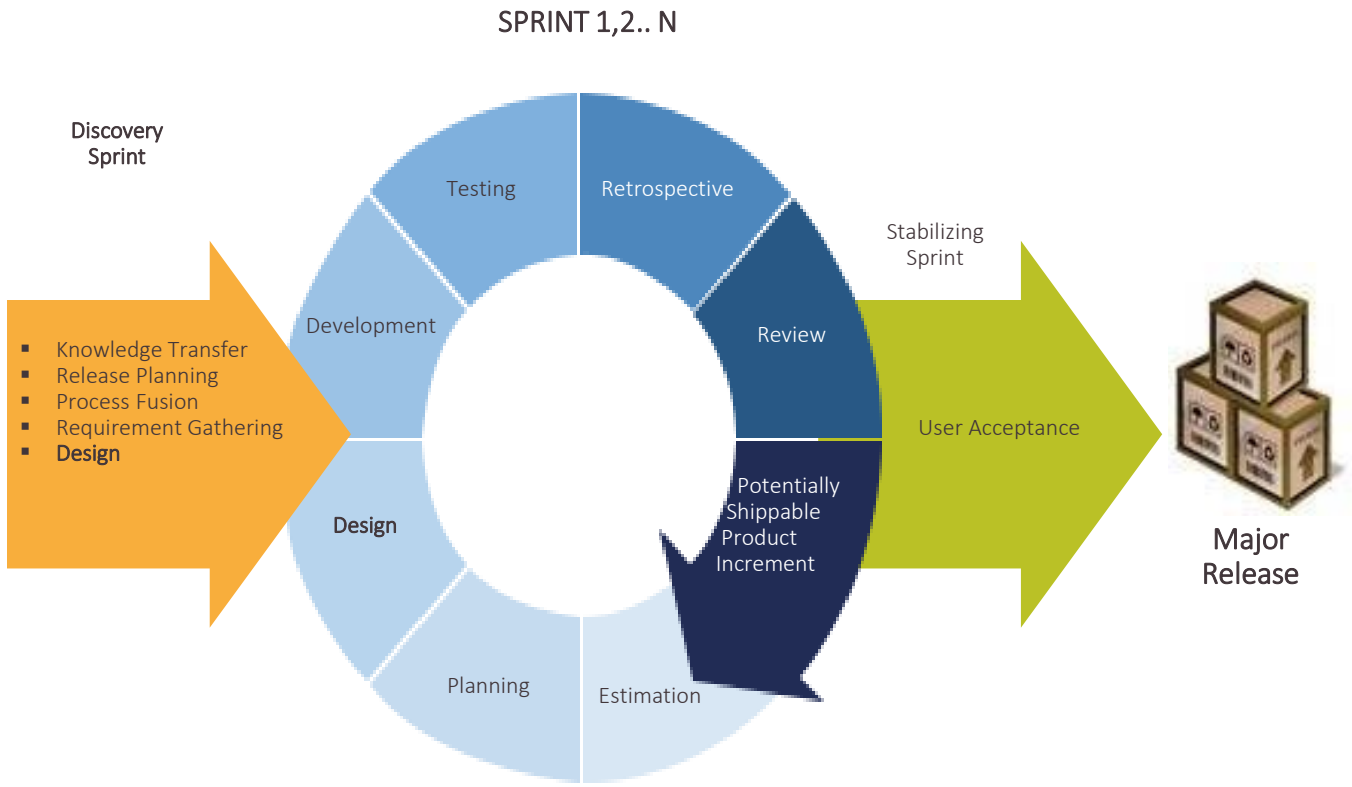
# Delivery Model…

- Project execution

Agile Scrum model defines the framework in which all activities of architecture and development will take place using the fundamental principles of Scrum. Agile Scrum model is the best model to follow considering the complexities, scale and distributed nature of the “One Unified App”. It will help in giving the periodic deliveries to organization for their review and an early opportunity to make the changes, if needed.

Project Management portal for task management and reporting and planning purpose. The following phases will be considered as indicated below:

- Discovery
- Development
- Stabilization ( Integration and System testing)
- User Acceptance
- Major Release/Milestone Deployment
- Others – Knowledge Transfer and
- Post production support, if required



# Delivery Model…

- Project execution: Development

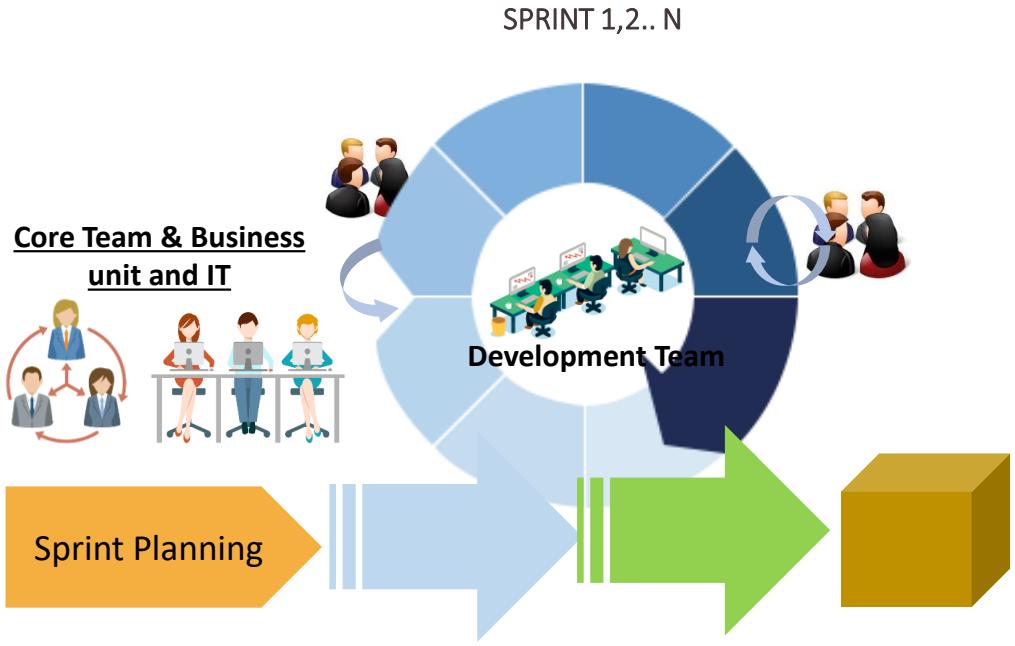
Development will be carried out in Agile Scrum approach having sprint designed of 2-3 weeks duration depending upon the size of backlog item which will provide ready to test. These sprints output will ensure speedy development and visible result for stakeholders and would assure the on time and expected deliverable of “API”.

The scope of these sprints will be decided in “Sprint Planning” meeting jointly by organization, Business and Core team and would ensure that business priority will be met as per expectation.

This phase would ensure the development, design as well as internal testing of the scope of work, that the team is committed to deliver.

**Organization Business and POC**

| | User Stories |
|---|---|
| 1 | X Feature |
| 2 | Y Feature |
| 3 | A Feature |
| 4 | B Feature |
| 5 | … |
| 6 | ….. |
| | |




Topic 1 : X Feature
Topic 4 : B Feature
……

Sprint Retrospective with scrum team

Thank you!