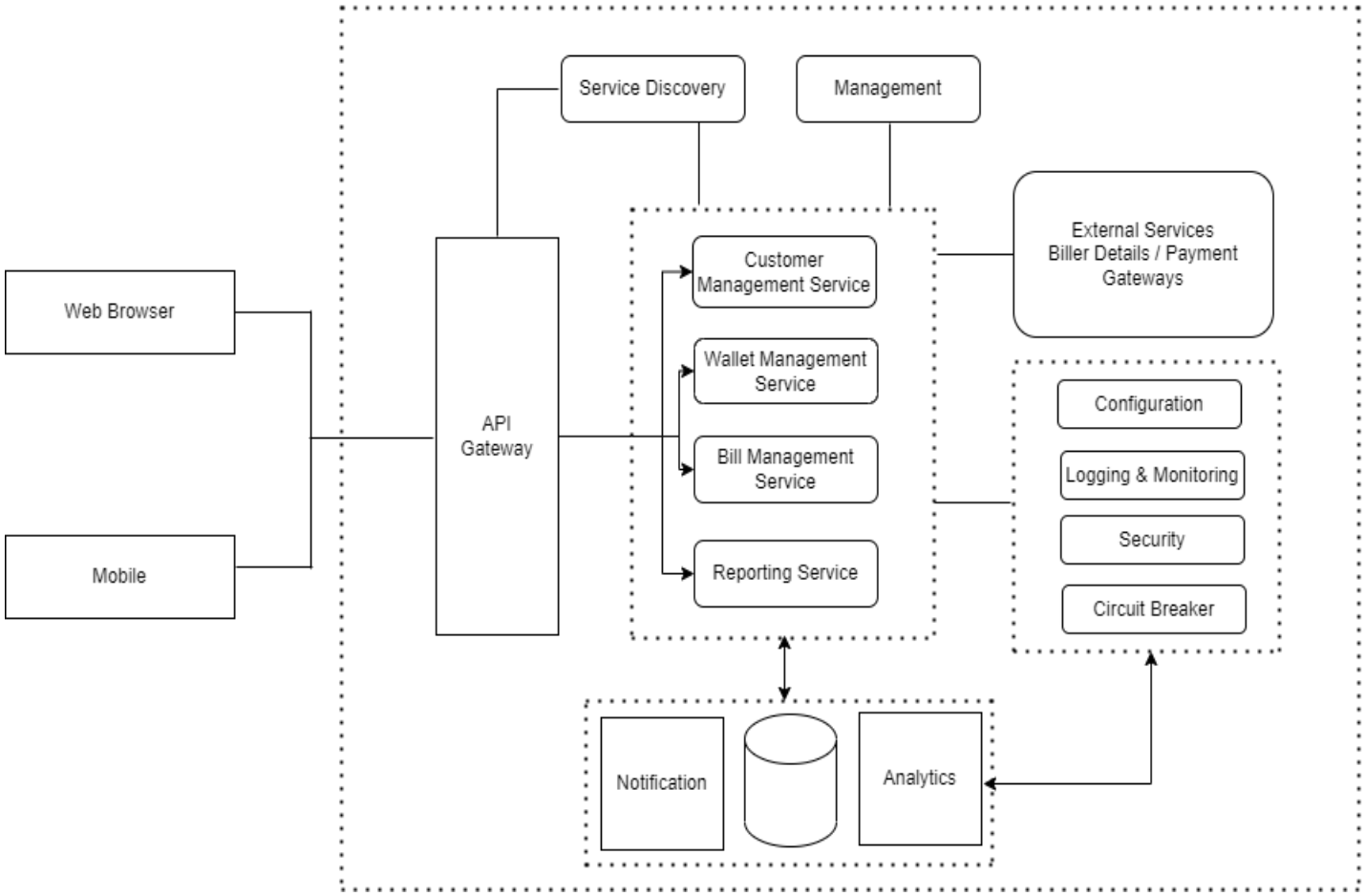
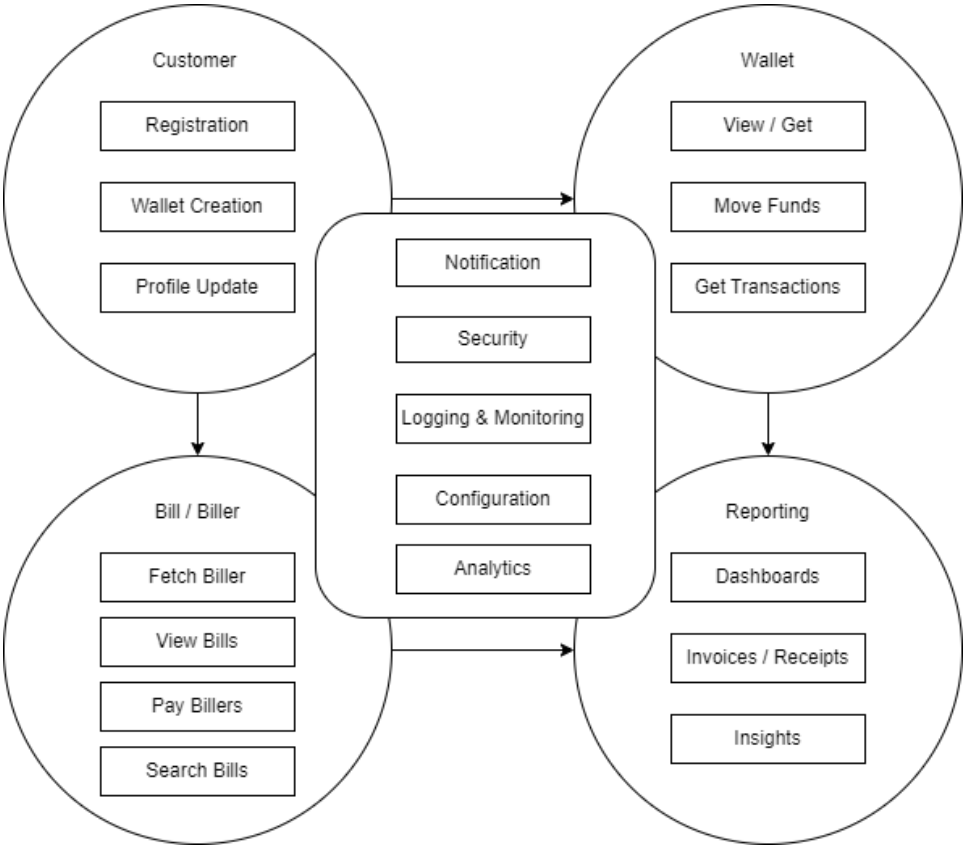
XYZ Bill Pay Application

- Proposed High Level Architecture



- Bounded Context Diagram



- Component Break-up
    - API Gateway – This layer acts as first level of abstraction and protects direct exposure of REST services to external and third-party services. Key benefits for gateways are as follows,
        - Security
            - Network - Firewall rules including cloud policies
            - Centralized Authentication and Authorization - IAM
        - Routing, limiting access and service aggregation
        - API versioning, logging and health monitoring
        - Load balancing and caching

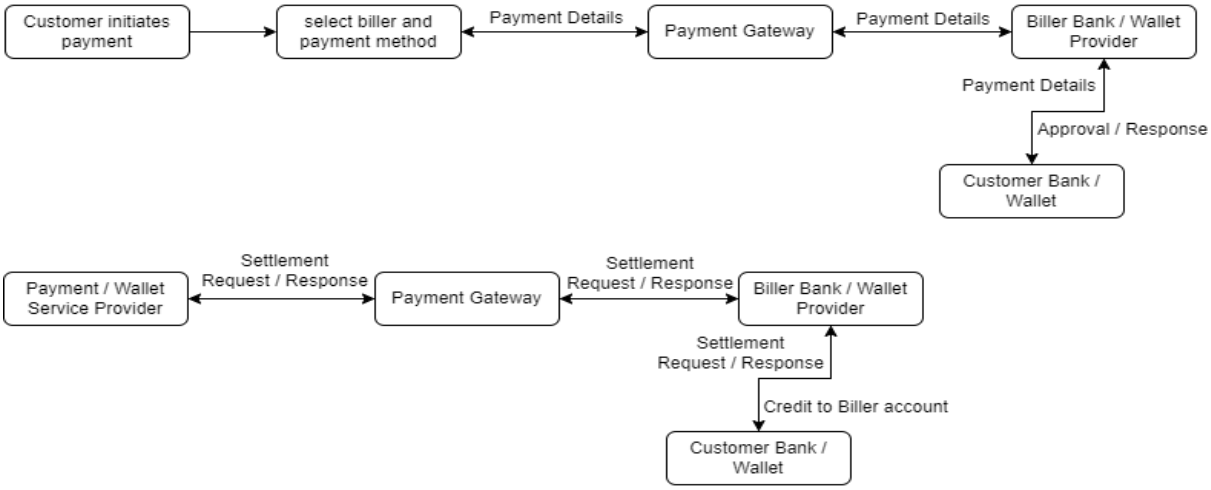
- Microservices
  - **Customer Management** – This is customer specific service and will help with,
    - Registration including wallet creation/assignment on success
    - Login/Logout with help of password protection policies
    - Profile update
    - Changing password and PIN for wallet transactions
  - **Wallet Management** – This service is for managing actions related to customer wallet,
    - Move funds within wallet account
    - Get / View wallet account
    - Get transactions
  - **Bill / Biller management** – This provides an interface to customer to manage their utility bills and perform desired actions,
    - View Bills
    - Search bills
    - Make payments
    - Support for bulk payment process
  - **Reporting service**
    - Generating transactions reports
    - Generating invoices/receipts of payments
    - Ability to search specific transactions
    - Providing insight full dashboards to customer to see different views

- Common Services / Utilities – System provides all important and critical services below,
  - Layered security for every microservice – data encryption for sensitive data storage and transfer across microservices
  - Logging and monitoring which helps to identify issues to improve system
  - External configuration adding loose coupling and can be managed without impacting application builds
  - Circuit breakers to avoid complete system failures, isolating failures to only MS having issues
  - Analytics support to detect patterns for un-usual activities, preventing frauds, to get insights to serve customer in better way

- Service Discovery and management – These services help to manage different microservices,
  - Including their lifecycle
  - handling auto-scaling
  - fault tolerance
  - high availability
  - Service Registry and discovery

- Database Management
  - Data management is critical and most important part of any application
  - Depending on functionality and use cases we can use different database systems such relational, document based, object based etc.
  - We can choose to have one of the below option while designing microservice
    - Single service database
    - Shared service database
  - If there is clear separation of domains it's better to have separate database for every service. But if there is tight dependency between domains it's better to have single database.
  - Avoid to have distributed transactions until it's really required, as it adds complexity and risks
  - Also, need to consider data security, compliance and cost while planning to store on cloud

- External System Integration -
  - External service integration is required to fetch biller list and bill details for customers
  - Payment gateway integration also required to transfer money between customer wallet to biller account
  - Simple flow for money transfer when user makes a payment against the bill

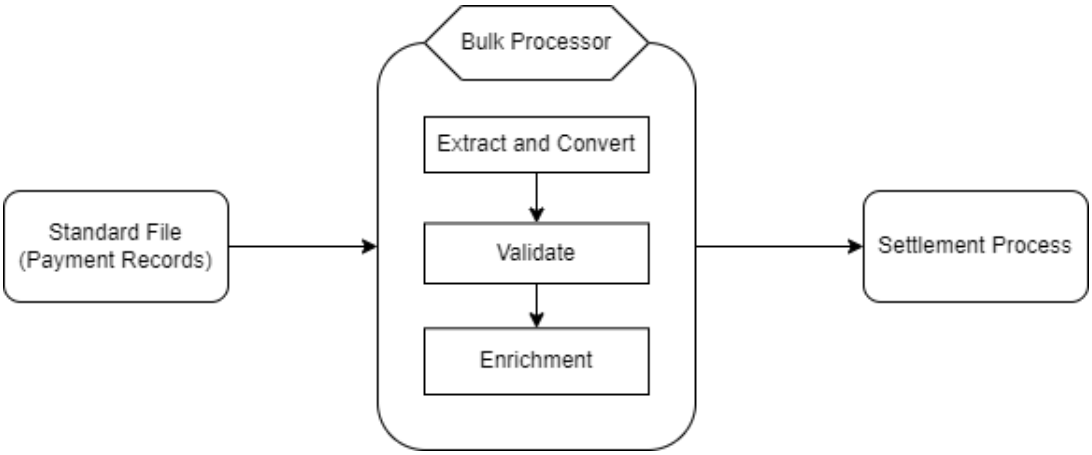


- Notification Service:
  - This service is used to send alert and messages to customer regarding wallet account balance, bill details, transaction details, promotional and discount offers

- Bulk Bill Payment Service:
  - This service will help customers to process bill payments in bulk by providing simple files with records
  - Standard file format is as follows and all customer should adhere to it

User Id / Name	Bill Id	Wallet Id	Amount

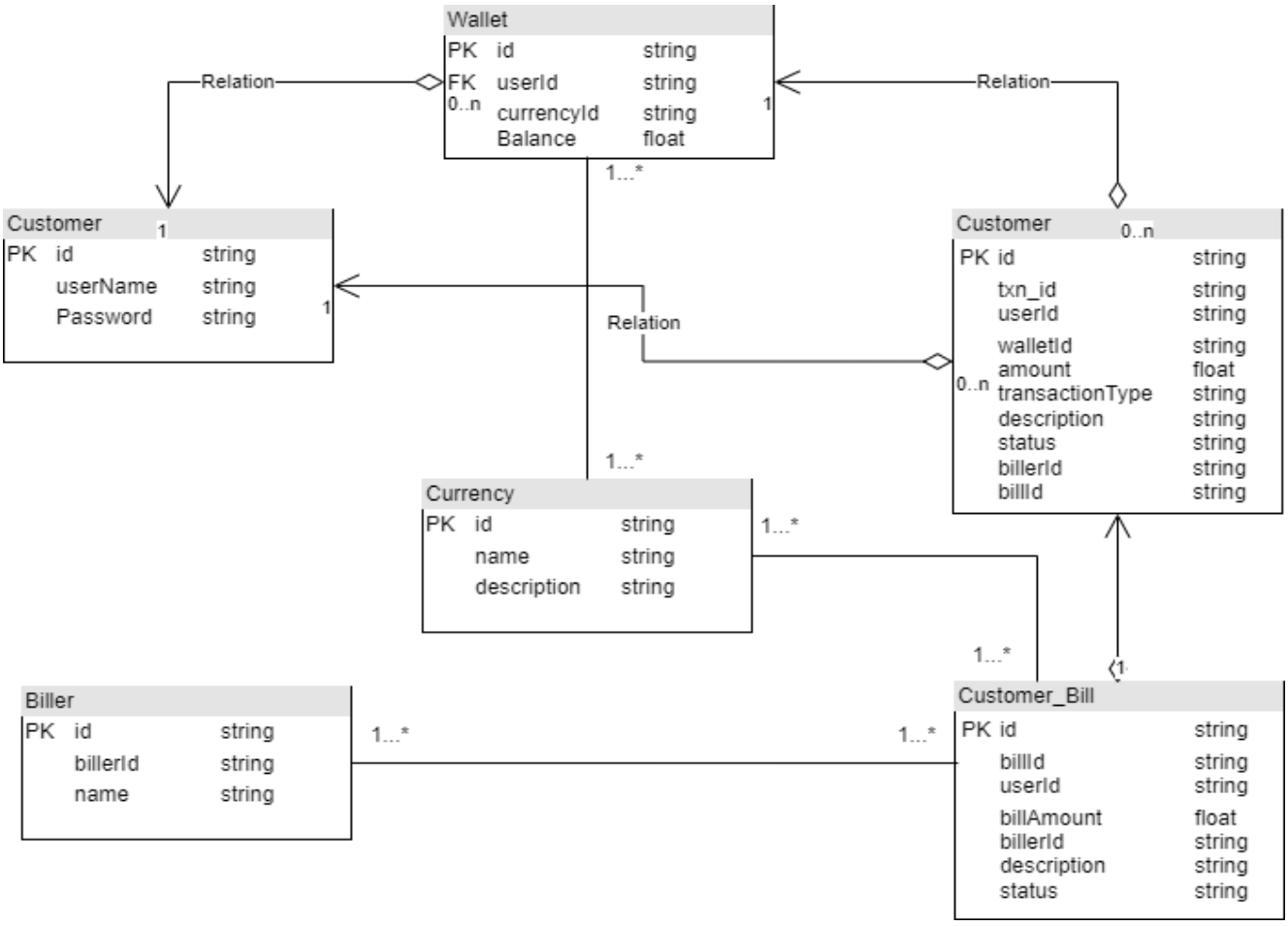
  - Bulk process flow – Settlement process is same as shown in external integration diagram (refer 2nd flow)



- Testing Strategies
  - Unit Tests
  - Contract Tests
  - Integration Tests
  - Component Test
  - End to end test

- Technology Stack
    - Development Technologies – Java, Spring Framework-Boot-Cloud, Hibernate/Spring Data JPA, Angular/React, Talend (ETL), Maven
    - Data Management –
        - Cloud Agnostic - RDS (MySQL, Oracle, PostgresSQL, File storage (Mongo), Object Storage (HDFS)
        - Cloud Native (AWS) – RDS (Aurora), Document DB, Object Storage (S3), DynamoDB
    - In-Memory/Cache
        - Cloud Agnostic – Redis, Hazelcast
        - Cloud Native – Elastic cache
    - CICD
        - Cloud Agnostic – Jenkins, Github, BitBucket
        - Cloud Native (AWS) - CodePipeline (CodeCommit, CodeBuild, CodeDeploy)
    - Containers
        - Cloud Agnostic – Docker, Kubernetis, Openshift
        - Cloud Native (AWS) - ECS, EC2, Lambdas
    - Notification
        - Cloud Agnostic – Kafka, ActiveMq
        - Cloud Native (AWS) - SNS, SQS
    - Reporting and Analytics
        - Tableau, Cognos
    - Monitoring and Logging
        - Cloud Agnostic – ELK, Splunk, Grafana, Prometheus
        - Cloud Native (AWS) - CloudTrail, CloudWatch
    - Automation Test Suits
        - Junits, Mockito, Selenium, JMeter
    - API Gateway
        - Zuul, Spring Cloud Gateway, Ribbon (Load Balancer), Eureka (Naming server), Hystrix (Fault Tolerance), Zipkin (Distributed Tracing)
    - Security – Firewall, Security Groups and policies, SSL, TLS, JWT/CSRF

- Data Model


Wallet
PK id string
FK userId string
currencyId string
Balance float
0..n
1
Relation
Relation
1...*
Customer
1
PK id string
userName string
Password string
1
Relation
Customer
0..n
PK id string
txn_id string
userId string
walletId string
amount float
transactionType string
description string
status string
billerId string
billId string
0..n
1...*
Currency
PK id string
name string
description string
1...*
1...*
Biller
PK id string
billerId string
name string
1...*
1...*
Customer_Bill
PK id string
billId string
userId string
billAmount float
billerId string
description string
status string

- API and Specification
    - Customer registration

      API - /bill-pay-app/register

      {
      “userId”: “new-user”,
      “Currency”: “INR”
      }
    - Move Funds within wallet

      API - /bill-pay-app/move-funds

```
{
“userId”: “userId”,
“currency”: “INR”,
                “walletId”: "2",
“amount”:2000,
             “transactionType”: “C”,
                “comments”: “add money”
}
```

    - Get wallet by user id

      API - /bill-pay-app/user?userId={user}

    - Get transactions by wallet id

      API - /bill-pay-app/wallet/{id}/transactions

    - View bills

      API - /bill-pay-app/user/{id}/bill

    - Search bill

      API - /bill-pay-app/user/{id}/bill/{id}

    - Pay Biller

      API - /bill-pay-app/pay-bill

```
{
“userId”: “userId”,
“currency”: “INR”,
                “walletId”: “wallet-id”,
“amount”: “2000”,
             “billId”: “bill-id”
                    “billerId”: “biller-id”,
             “comments”: “Electricity Bill paid”
}
```