**Bill Pay Application**

**Overall System Representation**

* Overall system representation present in **overall-system-representation.jpg**

**Data Model**

* Data Model is explained in **data\_model.jpg**

**Micro services**

1. Registration-ms
2. Wallet-ms
3. Billing-ms
4. Static-data-ms
5. Bill-payment-ms
6. Bulk-payment-ms
7. Scheduler-ms
8. Transaction-ms
9. Settlement-ms

**Detail Design**

* Detail design present in **detail-microservice-level-flow-design.jpg**

**Micro-Service Details**

**1) Registration-ms:**

Responsibilities:

* Take input as email-id. Make entry in Customer table
* Push the msg on topic to create wallet for that customer
* Database- RDBMS

**2) Wallet-ms:**

Responsibilities:

* Listen msg on topic and create wallet for that customer by making entry in wallet table with zero initial balance
* provide add money feature

When user enter amount and select payment method and enter details

* 1. Start transaction. get transaction id from transaction-ms
  2. Securely give call to payment gateway ,depending on response update wallet balance
  3. Push msg on topic to log transaction
  4. Save payment details in db in PCI\_DSS guideline
* Database- RDBMS

**3) Billing-ms:**

Responsibilities:

* Give call to static-data-ms to fetch registered vendors and their required fields.
* Fetch bill feature- to fetch bill for registered vendors from external.

**4) Static-data-ms:**

Responsibilities:

* It holds the registered vendor details.
* Cache: Redis

**5) Bill-payment-ms:**

Responsibilities:

* To fetch bill from billing-ms
* Fetch wallet balance from wallet-ms
* validate bill against wallet-ms
* Get transaction id from transaction-ms and start transaction
* Deduct money from wallet
* pay bill using external vendor site
* persist bill payment transaction details and status
* If bill payment is failed by auto retry then initiate settlement by pushing those transactions on settlement topic.

**6) Bulk-payment-ms:**

Responsibilities:

* Parse the billing file provided by user using relevant file parser (factory pattern).
* map the template according to file type.
* Persist each billing record in a staging table as a row in pending status.
* Staging Database - RDBMS

**7) Scheduler-ms:**

Responsibilities:

* It is fixed time delay recurring spring job
* Read each pending record from staging table
* Facilitate bill payment by giving call to Bill-payment-ms to pay bill.
* Depending upon response of Bill-payment-ms update status in staging table.
* Once all records in the file are processed then mark that file as a complete.

**8) Transaction-ms:**

Responsibilities:

* log transaction details.
* Listen wallet transaction-topic.
* Make entry in wallet\_transaction table.
* Database- RDBMS

**9) Settlement-ms:**

Responsibilities:

* Settle external failed payment transactions.
* Bill-payment-ms push the message on settlement topic and settlement-ms listen those messages.
* Invoke wallet-ms apis to credit earlier debited amount (compensating transactions).

**Technology Stack:**

* Language: Preferred Java spring boot (as its suited and well versed to it)
* Database: preferred RDBMS
* Service discovery: Eureka
* Load balancer: Nginx
* Cache: Redis
* Messaging: kafka
* API Gateway: Kong
* Container: Docker
* Container Orchestration: Kubernates
* Security: TLS
* Distributed tracing: Zipkin
* Monitoring: Promentheus & Graphana, jprofiler

**Testing:**

1. Acceptance: BDD
2. Dev: Junit
3. Load testing: jmeter

**CICD:**

1. Jenkins and bitbucket
2. Gitlab

**External and Internal calls:**

1. External calls redirect through firewall and proxy configuration.
2. Once request came through firewall and proxy and also authentication, authorization have been carried out. Internal ms to ms communication falls under green zone. These calls are direct calls through registered service name.